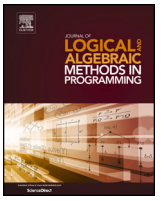
J. Log. Algebraic Methods Program. 140 (2024) 100978

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

journal homepage: www.elsevier.com/locate/jlamp



Succinct ordering and aggregation constraints in algebraic array theories

Rodrigo Raya [*,1], Viktor Kunčak

School of Computer and Communication Science, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

ARTICLE INFO

Keywords:
Decision procedures
Satisfiability modulo theories
Arrays
Feferman-Vaught
Composition theorems

ABSTRACT

We discuss two extensions to a recently introduced theory of arrays, which are based on considerations coming from the model theory of power structures. First, we discuss how the ordering relation on the index set can be expressed succinctly by referring to arbitrary Venn regions. Second, we show how to add general aggregators to the calculus. The result is a logic that subsumes four previous fragments discussed in the literature and is distinct from array fold logic, in that it can express summations, while its satisfiability problem remains in non-deterministic polynomial time.

1. Introduction

Motivated by the problem of specifying and verifying programs manipulating arrays data structures, we proposed in [59] an extension of combinatory array logic [16] that can handle an ordering relation on the index set and can handle cardinality constraints on the set of indices. This extension subsumes several array theories proposed in the literature [66,16,25,2] and we showed that it can be integrated in SMT solvers [17,6] and verifying compilers [31,39,68,70,74] thanks to its closure properties under propositional operators and Hoare logic triples.

The key idea in designing the fragment discussed in [59] is that the vector operations that occur in combinatory array logic have been previously studied in algebra in the setting of products of structures and their model theory has been examined by many authors starting by Mostowski [45], Feferman [64] and Feferman-Vaught [22]. However, the results of Feferman and Vaught consider fully quantified first-order theories, and it was not clear whether similar decompositions could be done for prefix fragments of first-order theories. [59] gives a positive answer to this question in the case of existential fragments of generalised powers, where the underlying index set is arbitrary and the components of the structure are assumed to be identical.

The goal of this paper is two-folded.

First, we revisit the addition of an ordering relation to the index structure and show that unlike in [59] this can be done succinctly with an automaton or regular expression over propositional letters rather than bit-strings. Similar automata have been studied in the theory of symbolic automata [72] and until recently, it was not clear how to compute the Parikh image [49] of their language without incurring in an exponential blow-up [26,57]. We also show that this technique is applicable to other data structures, such as trees. More importantly, the challenge of computing the Parikh image of such an automaton reveals another connection with [22, Theorem 3.1]: the need to assume that the sequence of sets used is a partitioning sequence.

* Corresponding author.
E-mail addresses: rodrigo.raya@epfl.ch (R. Raya), viktor.kuncak@epfl.ch (V. Kunčak).

[1] Research supported by the Swiss NSF Project P500PT_222338.

https://doi.org/10.1016/j.jlamp.2024.100978
Received 24 November 2023; Received in revised form 21 February 2024; Accepted 10 May 2024

Available online 14 May 2024

Second, we discuss the addition of aggregation operators to the language in [59]. While the language in [59] already contained a cardinality operator, capable of counting the number of components of the array that satisfy certain property, its role was mainly that of allowing the combination of the theories of ordering and interpreted sets. In this paper, we also consider other aggregation operators which appear in applications both in verification [13] and database querying [28]. In particular, we discuss the minimum, maximum, and summation operators. In particular, the introduction of the summation operator, lifts the restriction mentioned in [16] that the cardinality of multisets cannot be expressed. For all the newly introduced aggregation operators, we show that the computational complexity of deciding the resulting theory is NP-complete whenever the theory of the elements is in NP, which is usually the case in satisfiability modulo theories.

As a result, we obtain a modular framework in which to express a family of array theories that have been studied in the literature [66,16,25,2] and produce a theory that is distinct in terms of expressive power with [13], while retaining an NP upper bound on the computational complexity of the satisfiability problem.

Organisation of the paper. The rest of the paper is organised as follows. In Section 2 we recall the Feferman-Vaught theorem for product structures, our instantiation for power structures in [59] and motivate the problem of including succinct ordering relations in the specification language. In Section 3 we describe the techniques to introduce the succinct ordering without affecting the computational complexity of the problem. In Section 4, we show that the technique is equally applicable to tree data structures, although it needs related results of the literature. In Section 6, we add the minimum and maximum operators to the language. In Section 7, we add the summation aggregator. In Section 8, we discuss the resulting classification of array theories from the literature. In Section 9, we discuss how to encode the examples presented in [13] and supplementary examples expressible in our fragment. Section 10 concludes the paper.

2. Preliminaries

We recall next some concepts from logic and computational complexity which make the paper self-contained. For further details, the reader may consult [29,4].

2.1. First-order logic

The formal theories we study are written using a first-order language.

Definition 1. A first-order language is one whose logical symbols are $\neg, \wedge, \vee, \forall$ and $\exists$, whose terms are either variables, constants or function symbols applied to terms and whose formulas are either atomic (relation symbols applied to terms) or general (atomic formulas and inductively, from formulas A, B, we get new formulae $\neg A$, $A \wedge B$ and $A \vee B$ and from a formula A and a variable symbol x we get the new formulae $\exists x.A$ and $\forall x.A$).

Definition 2. A variable in a formula is free if there is no occurrence of that variable under a quantifier on the path of the syntax tree of the formula reaching the occurrence of the variable. A formula without free variables is a sentence.

It is customary to study the computational properties of theories in terms of the quantifiers used. To this end one introduces the prenex form of formulae [61].

Definition 3. A prenex normal form of a first-order formula F consists of a string of quantifiers (called the prefix of the formula) followed by a quantifier-free formula (known as the matrix of the formula) which is equivalent to F.

The computational properties of logical theories have been the object of intense research since the pioneering work described in [23]. This research showed that traditional decidable theories studied by logicians were strongly intractable in practice. For the sake of example, [65] proved that a circuit deciding the weak monadic second-order theory of one successor would not fit in the known universe.

The computational complexity of a logical theory usually increases with the number of quantifier alternations. For this reason, in applications one often focuses on the existential fragment of theories: the subset of formulae of a theory whose prefix is made of existential quantifiers only. This is the case for the applications in program verification that are considered in this work [31,7,35].[2]

More precisely, we will study existential fragments of theories defined semantically in terms of the models that the formulae in the theory satisfy.

Definition 4. A structure $\mathcal{A}$ over a first-order language L is a tuple with four components: a set A called the domain of A; a set of elements of A corresponding to the constant symbols of L; for each positive integer n, a set of n-ary relations on A (i.e. subsets of

[2] Note, however, that important theoretical research questions, such as the existence of quantifier-elimination procedures for linear arithmetic with summation constraints and its computational complexity for arbitrary quantifier alternations remain open. We are aware of a group of researchers at the University of Oxford currently working on this direction.

A^n), each of which is named by one or more n-ary relation symbols of L and for each positive integer n, a set of n-ary operations on A (i.e. maps from A^n to A), each of which is named by one or more n-ary function symbols of L. The mapping assigning each first-order symbol of L to its corresponding interpretation in $\mathcal{A}$ is denoted $\cdot^{\mathcal{A}}$. This function is extended to work on terms, i.e. the application of function symbols to constants, variables or other terms, by requiring that

$$(f(s_1, \ldots, s_n))^{\mathcal{A}} = f^{\mathcal{A}}(s_1^{\mathcal{A}}, \ldots, s_n^{\mathcal{A}})$$

We define the notion of satisfaction relation restricted to sentences.

Definition 5. Let ϕ be a sentence in a first-order language L and let $\cdot^{\mathcal{A}}$ be an interpretation of the symbols of L in the structure $\mathcal{A}$. The sentence ϕ is satisfied in the structure $\mathcal{A}$, written $\mathcal{A} \vDash \phi$, if the following conditions apply.

- If ϕ is the atomic sentence $R(s_1, \ldots, s_n)$ where $s_1, \ldots, s_n$ are terms of L then $\mathcal{A} \vDash \phi$ if and only if $(s_1^{\mathcal{A}}, \ldots, s_n^{\mathcal{A}}) \in R^{\mathcal{A}}$.
- $\mathcal{A} \vDash \neg\phi$ if and only if it is not true that $\mathcal{A} \vDash \phi$.
- $\mathcal{A} \vDash \phi_1 \wedge \phi_2$ if and only if $\mathcal{A} \vDash \phi_1$ and $\mathcal{A} \vDash \phi_2$.
- $\mathcal{A} \vDash \phi_1 \vee \phi_2$ if and only if $\mathcal{A} \vDash \phi_1$ or $\mathcal{A} \vDash \phi_2$.
- If ϕ is the sentence $\forall y.\psi(y)$ then $\mathcal{A} \vDash \phi$ if and only if for all elements b of A, $\mathcal{A} \vDash \psi(b)$.
- If ϕ is the sentence $\exists y.\psi(y)$ then $\mathcal{A} \vDash \phi$ if and only if there is at least one element b of A such that $\mathcal{A} \vDash \psi(b)$.

As announced, we will focus on the existential fragment of theories defined by some model.

Definition 6. The existential theory $Th_{\exists^*}(\mathcal{A})$ of a structure $\mathcal{A}$, is the set of existentially quantified formulas ψ of L such that $\mathcal{A} \vDash \psi$. A solution of the formula ψ is a satisfying assignment to its existential variables.

Because we focus on existential fragments, the computational problems we study are similar to problems in predicate clause logic or propositional logic [9]. Strictly speaking when determining membership in $Th_{\exists^*}(\mathcal{A})$ we are studying the validity problem for the existential fragment, while we are solving the satisfiability problem if we consider the same fragment as part of predicate clause logic.

More importantly, one can define normal forms for the matrix of the formulae in the studied fragment. The familiar notion of disjunctive normal form is used in the decision procedures below to reduce the satisfiability problem of the existential fragment to the satisfiability problem of a conjunction of literals. The notion of Stone normal form [30] is implicitly used when studying the corresponding theories with the cardinality operator on index sets. These are defined in a completely analogous way to the propositional case by forgetting the additional first-order structure.

2.2. Second-order logic

We study in this paper some theories that mix first-order logical expressions and quantification over sets of indices. To talk about these sets of indices it is convenient to introduce second-order languages.

Definition 7. A second-order language is one that allows existential and universal quantification over relations. A second-order language is monadic when second-order quantification is restricted to sets, i.e. unary relations.

In this work, we will be concerned mostly with monadic theories of structures and with interpretation of the second-order symbols that are restricted to finite sets, so-called weak interpretations.

Definition 8. The weak monadic second-order theory $\mathrm{WMSO}(\mathcal{A})$ of a structure $\mathcal{A}$, is the set of monadic formulas ψ of L such that $\mathcal{A} \vDash \psi$ when set variables are interpreted over finite sets.

We give some examples that will appear in later sections.

Example 9. The weak monadic second-order theory of one successor is the theory $\mathrm{WS1S} = \mathrm{WMSO}(\langle \mathbb{N}, +1 \rangle)$ where $+1$ is the function that adds one to each natural number.

Example 10. The theory of Boolean algebra with Presburger arithmetic is the theory $\mathsf{BAPA} = \mathrm{WMSO}(\langle \mathbb{N}, \subseteq, \sim \rangle)$ where $\subseteq$ is the subset relation between sets and $\sim$ is the equicardinality relation between sets.

Note that BAPA can be presented as a first-order theory with domain $\mathcal{P}(\mathbb{N})$ as in [36] or as a second-order theory as it is done here. We choose the second-order approach because we will describe in this work extensions of BAPA that use first-order features. [36, Section 2] shows how to encode integers and Boolean algebra operations with the operators $\subseteq$ and $\sim$.

2.3. Feferman-Vaught theorem

As a motivation for the first half of the results in the paper, we recall next the Feferman-Vaught theorem. The theorem solves the decision problem for various product structures usually found in algebra. An example of these is the direct product (see [22, Section 4] for more examples).

Example 11 *(Direct product).* The direct product of a non-empty indexed family of structures $\mathfrak{A} = \left\langle \mathfrak{A}^{(i)} \mid i \in I \right\rangle$, where for each $i \in I, \mathfrak{A}^{(i)} = \left\langle A^{(i)}, R^{(i)} \right\rangle$, is the structure $\langle D, U \rangle$, where

$$D = \mathcal{P}\left(A^{(i)} \mid i \in I\right)$$

is the set of functions with domain I and such that for each $i \in I$, the image of i is in $A^{(i)}$ and for any $g, h \in D, \langle g, h \rangle \in U$ if and only if, for every $i \in I, \langle g(i), h(i) \rangle \in R^{(i)}$.

In the case that all the structures $\mathfrak{A}^{(i)}$ are equal, the resulting structure is called a direct power. For a familiar case, the reader may think of the direct power of the structure $(\mathbb{R}, +)$ to obtain structures of the form $(\mathbb{R}^n, +)$ where the addition in the power structure is done component-wise.

It turned out, for the purpose of analysing the decision problem of all these structures, that it was more convenient to define the notion of generalised product, which we recall next.

Before introducing the notion of generalised product, we need to formalise the notion of set interpretation [22, Definition 2.1], which was heavily used in [59].

Definition 12 *([22]).* Suppose that $\mathfrak{A} = \left\langle \mathfrak{A}^{(i)} \mid i \in I \right\rangle$ is an indexed family of relational systems $\mathfrak{A}^{(i)} = \left\langle A^{(i)}, \dots \right\rangle$ using a first-order signature μ and $D = \mathcal{P}\left(A^{(i)} \mid i \in I\right)$ be the set of functions mapping each index $i \in I$ to a value in $A^{(i)}$. Let $f = \langle f_0, \dots, f_n, \dots \rangle \in D^\omega$, and let θ be a formula of the first-order language over the signature μ. Then we put

$$K_\theta^{\mathfrak{A}}(f) = \left\{ i \mid i \in I \text{ and } \mathfrak{A}^{(i)} \vDash \theta(f(i)) \right\}$$

where $f(i) = \langle f_0(i), \dots, f_n(i), \dots \rangle$ and $f_j(i)$ denotes the i-th component of the vector f_j. In particular, if θ is a sentence, $K_\theta^{\mathfrak{A}}[f]$ does not depend on f, and we write $K_\theta^{\mathfrak{A}} = K_\theta^{\mathfrak{A}}(f) = \left\{ i \mid i \in I \text{ and } \mathfrak{A}^{(i)} \vDash \theta \right\}$.

With the notion of set interpretation in place, we are ready to introduce generalised products [22, Definition 2.4].[3]

Definition 13 *([22]).* Suppose that $\mathfrak{A} = \left\langle \mathfrak{A}^{(i)} \mid i \in I \right\rangle$, is a non-empty indexed family of relational systems $\mathfrak{A}^{(i)} = \left\langle A^{(i)}, \dots \right\rangle$, and $D = \mathcal{P}\left(A^{(i)} \mid i \in I\right)$, and let $\mathfrak{S} = \langle \mathcal{P}(I), \dots \rangle$ be an algebra of subsets of I. For each standard acceptable sequence $\zeta = \langle \Phi, \theta_0, \dots, \theta_m \rangle$, with p free variables, we put

$$Q_\zeta^{\mathfrak{A},\mathfrak{S}} = \left\{ \left\langle f_0, \dots, f_{p-1} \right\rangle \mid f \in D^\omega \text{ and } \mathfrak{S} \vDash \Phi\left[K_{\theta_0}^{\mathfrak{A}}(f), \dots, K_{\theta_m}^{\mathfrak{A}}(f)\right] \right\}$$

By the generalised product $\mathcal{P}(\mathfrak{A}, \mathfrak{S})$ of the systems $\left\langle \mathfrak{A}^{(i)} \mid i \in I \right\rangle$ relative to the algebra $\mathfrak{S}$ of subsets of I, we mean the system

$$\mathfrak{D} = \left\langle D, Q_{\zeta_0^0}^{\mathfrak{A},\mathfrak{S}}, \dots, Q_{\zeta_n^0}^{\mathfrak{A},\mathfrak{S}}, \dots \right\rangle.$$

In the above definition, the sequence ζ is acceptable, which means that the formulas $\theta_0, \dots, \theta_m$ are first-order formulas on the structures $\mathfrak{A}^{(i)}$ while the formula Φ is a first-order formula on structures of a class of algebras of subsets of the form

$$\mathfrak{S} = \left\langle \mathcal{P}(I), \emptyset, \cup, \cap, -, \subseteq, M_1, \dots, M_j, \dots \right\rangle$$

such that I is any non-empty set and $\emptyset, \cup, \cap$ and $\subseteq$ are the usual set theoretic operations or relations restricted to $\mathcal{P}(I)$ (the set of subsets of I).

Naturally, the first-order formulas on $\mathcal{P}(I)$ correspond to a second-order formula on I. Some of the structures that are considered in [22] include the equicardinality relation $\sim$, the finiteness predicate Fin and a well-ordering relation $\prec$. The result of Feferman-Vaught implies that denoting by $Th(\cdot)$ the set of first-order sentences that are true on a certain structure, then deciding $Th(\mathfrak{D})$ reduces to deciding the theories $Th(\mathfrak{A}^{(i)})$ for each $i \in I$ and the theory $Th(\mathfrak{S})$ [22, Theorem 3.1].

[3] The notion of generalised products may appear at first somewhat obscure. Feferman and Vaught noted: "It would be difficult to try trace the growth of ideas which led to the notion of the generalised product and to the statement and proof of the basic theorem 3.1 (...) However, something concerning the order of discoveries can at least be mentioned now." It may help the reader to consult [58,59] in the simpler case of direct powers to understand how the defined set interpretations arised in our work.

$$
\begin{aligned}
F &::= A \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F \\
A &::= B_1 = B_2 \mid B_1 \subseteq B_2 \mid T_1 = T_2 \mid T_1 \leq T_2 \mid K \text{ dvd } T \\
B &::= x \mid \emptyset \mid \mathcal{U} \mid B_1 \cup B_2 \mid B_1 \cap B_2 \mid B^c \\
T &::= k \mid K \mid T_1 + T_2 \mid K \cdot T \mid |B| \\
K &::= \ldots \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid \ldots
\end{aligned}
$$

Fig. 1. QFBAPA's syntax.

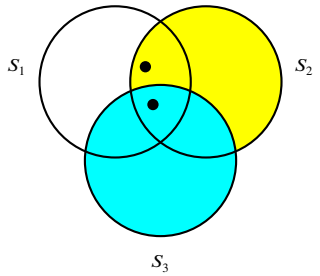


Fig. 2. An example Venn region diagram using set variables S_1, S_2, S_3. An example Venn region in blue is S_3 and in yellow is $S_2 \cap S_3^c$. The elementary Venn regions marked with dots are in blue $S_1 \cap S_2 \cap S_3$ and in yellow $S_1 \cap S_2 \cap S_3^c$. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

2.4. Known results about the algebra of indices

The analysis of the decidability of the set of sentences $Th(\mathfrak{S})$ is central for the Feferman-Vaught theorem. In this section, we focus on the preliminaries for the existential fragment of the theory $Th(\mathfrak{S})$.

2.4.1. Algebras with equicardinality relation

The theory of the structure

$$\langle \mathcal{P}(I), \emptyset, \cup, \cap, -, \subseteq, \sim \rangle \tag{1}$$

was studied profusely by Kunčak and his collaborators [36,38] under the name BAPA and QFBAPA in the quantifier-free fragment. In this paper, we will be mostly concerned with the fragment QFBAPA which can be defined as in Fig. 1.

In the grammar given in the figure, F presents the Boolean structure of the formula, A stands for the top-level constraints, B gives the Boolean restrictions and T the Presburger arithmetic terms and $\mathcal{U}$ represents the universal set. Boolean and integer variables are denoted with lowercase letter x and k respectively. Some relations like the divides relation dvd are added in order to preserve the expressivity of the fully quantified fragment (i.e. these are the relations one adds so that the full theory BAPA admits quantifier elimination). The semantics of operations in Fig. 1 is the standard one. We interpret integer terms as integers and interpret set terms as elements of the powerset of a universal set $\mathcal{U}$.

A fundamental concept in the decision procedures that we design is that of Venn region, which we now define formally.

Definition 14. Given a QFBAPA formula over the set variables $S_1, \ldots, S_k$, we define a Venn region as the equivalence class of Boolean algebra expressions on the variables $S_1, \ldots, S_k$ with respect to the relation of propositional equivalence. A Venn region is *elementary* if it (as an equivalence class) contains a Boolean algebra expression of the form $S_1^{l(1)} \cap \ldots \cap S_k^{l(k)}$ where $l(i)$ denotes the i-th bit in the bit-string letter and $S^0 := S^c$ (the complement of S) and $S^1 := S$.

Our procedure for checking satisfiability of extensions of QFBAPA works by reduction to QFBAPA and the proof of the equivalence between the original and the reduced formulas obtained by the procedure, relies on the argument [38] used to establish a complexity bound on the theory. This argument is based on a theorem by Eisenbrand and Shmonin [20] establishing a bound on the number of generators needed to express a given vector in their integer hull.

Definition 15. Given a subset $S \subseteq \mathbb{R}^n$, the integer conic hull of S is the set

$$int_{cone}(S) = \left\{ \sum_{i=1}^{t} \lambda_i s_i \middle| t \geq 0, s_i \in S, \lambda_i \in \mathbb{N} \right\}$$

Theorem 16 *(Eisenbrand-Shmonin). Let $X \subseteq \mathbb{Z}^n$ be a finite set of integer vectors, $b \in int_{cone}(X)$ and $M = \max_{\mathbf{x} \in X} \|\mathbf{x}\|_\infty = \max_{\mathbf{x} \in X} \max\{|x_1|, \ldots, |x_n|\}$. Then there exists a subset $X' \subseteq X$ such that $b \in int_{cone}(X')$ and*

$$|X'| \leq 2n \log(4nM)$$

The key idea that we will reuse for the correctness of our decision procedures is to reduce the cardinality constraints from each theory to a system of equations on the cardinalities of the elementary Venn regions determined by the Boolean algebra expressions. If $x_1, \ldots, x_k$ are the Boolean variables occurring in the Boolean expressions $b_0, \ldots, b_p$ under cardinalities in the formula, then the elementary Venn regions are of the form $p_\beta := \bigcap_{i=1}^{k} x_i^{e_i}$ for $\beta = (e_1, \ldots, e_k) \in \{0,1\}^k$ where $x^1 := x$ and $x^0 := \mathcal{U} \setminus x$. If $[\![b_i]\!]_{\beta_j}$ is the evaluation of b_i as a propositional formula with the assignment given in β and we define $l_\beta := |p_\beta|$, then $|b_i| = \sum_{j=0}^{2^k-1} [\![b_i]\!]_{\beta_j} l_{\beta_j}$, so $\bigwedge_{i=0}^{p} |b_i| = k_i$ becomes a system of linear equations

$$\bigwedge_{i=0}^{p} \sum_{j=0}^{2^k-1} \begin{pmatrix} [\![b_0]\!]_{\beta_j} \\ \vdots \\ [\![b_p]\!]_{\beta_j} \end{pmatrix} l_{\beta_j} = \begin{pmatrix} k_0 \\ \vdots \\ k_p \end{pmatrix}$$

This exponentially sized linear system is then turned into a polynomially sized system using Theorem 16. This is the essence of the proof of the membership of the satisfiability problem of QFBAPA in non-deterministic polynomial time.

Theorem 17 *([38]). The satisfiability problem of QFBAPA is in NP.*

2.4.2. Algebras with ordering relation

More complex algebras of indices have been treated. For instance, Wies et alii [75] devised a method of combining theories of ordering[4] with BAPA using the notion of Parikh image of a regular language [49], this combination would correspond to the theory of the structure

$$\langle \mathcal{P}(\omega), \emptyset, \cup, \cap, -, \subseteq, \sim, \prec \rangle \tag{2}$$

where ω denotes the set of natural numbers and $\prec$ denotes the ordering relation on singleton sets of natural numbers, i.e. $\{n\} \prec \{m\}$ if and only if $n < m$ (cfr. [22, Section 10]). One should note at this point the stark contrast between the decision procedures for the theory of the structure (1) which was known already to Skolem [63] by quantifier elimination into a quantifier-free elimination set and the procedure for the theory of the structure (2) which was known to Feferman and Vaught by the results later published by Ehrenfeucht [19] but whose explicit account in print have only appeared recently in [69]. Instead, the combination procedure of Wies et alii [75] draws on the work of Büchi [11] which essentially showed the decidability of the theory by quantifier elimination into finite automata (certain universally quantified formulas), thereby pioneering the so-called automata-logic connection.

To motivate the extension done in [59], which we improve in this paper, we next review this connection, which we phrase in terms of so-called regular expressions with large alphabets.

2.4.3. Regular expressions with large alphabets

Kleene introduced in [34] a theory of *regular sets of tables* to describe the behaviour of McCulloch-Pitts nerve nets. In his interpretation, the columns of these tables are indexed by the time unit and the rows by the neuron in the described system. A particular entry is set to one if and only if the corresponding neuron *fires* at the time indicated by its column index. Büchi showed [11,10] that under the interpretation that the rows correspond to sets of natural numbers and the entries are interpreted as membership in these sets, then the relations between the sets represented by the rows are exactly those definable in weak monadic second-order theory of one successor (WS1S).[5] In particular, [11, Corollary 1] gives an alternative proof to [22, Theorem 10.1] of the decidability of this theory.

Example 18. Consider the regular set of tables given by the expression $\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right)^*$. One possible table satisfying this expression would be

[4] The paper [75] goes far beyond this particular combination, but this particular case is enough for the purposes of this theorem.

[5] The observation of Büchi has been generalised. Some terminology uses the suffix "on words" to refer to this kind of interpretation of logical theories. It is also common to find the expression "MSOL on words" in this context since the proof of Büchi equating regular languages with WS1S is also applicable to the formalism of monadic second-order logic.

$$R ::= \begin{pmatrix} l_1 \\ l_2 \\ \cdots \\ l_k \end{pmatrix} \mid R_1 R_2 \mid R^* \mid R_1 \cup R_2$$

Fig. 3. Syntax of regular expressions over large alphabets.

Here $A = \{1,3,5,7\}$ and $B = \{2,4,6,8\}$. In general, the corresponding WS1S formula would specify the set of the first n odd natural numbers for A and the set of the first n even natural numbers for B.

Note that a minor difference between the interpretation of Kleene and that of Büchi is that the latter specifies an infinite *table* which is filled with strings of zeros in all columns but in a finite subset.

We enrich the theory of the indices of power structures by considering those relations definable in WS1S. In particular, we allow to express an order between the components. To this end, we will further combine our algebra of indices with the sets expressed in the formalism of Büchi, which we term regular expressions over large alphabets. Fig. 3 shows the syntax of such expressions. The expressions are taken over bit-string alphabet letters in the set $\{0,1\}^k$ for a fixed $k \in \mathbb{N}$.

2.4.4. A fragment combining the equicardinality and ordering relations

Taking into account the combination method of [75], the connection between finite automata and regular expressions sketched in the previous subsection, and the definition of generalised power for the existential fragment of first-order logic in [58], we formulated in [59] an existential fragment of the first-order theory of the structure (2) of the following form:

Definition 19. Consider conjunctions of formulas F in QFBAPA, regular expressions R over s letters with $1 \leq s \leq 2^k$ over the alphabet $\{0,1\}^k$ and existential first-order formulas φ_i with $1 \leq i \leq k$ referring the component structure, i.e. of the form

$$\begin{aligned} &\exists S_1, \ldots, S_k . F(S_1, \ldots, S_k) \wedge \\ &\quad \exists t_1, \ldots, t_k . \left((t_1, \ldots, t_k) \in R(l_1, \ldots, l_s) \wedge \bigwedge_{i=1}^{k} S_i = \{ n \in \mathbb{N} \mid t_i(n) \} \right) \wedge \\ &\quad \exists x_1, \ldots, x_m . \bigwedge_{i=1}^{k} S_i = \{ n \in \mathbb{N} \mid \varphi_i(x_1(n), \ldots, x_m(n)) \} \end{aligned}$$

where φ_i are formulae of a theory whose satisfiability problem is decidable in NP. We call this theory RegExp-QFBAPA-Power.

Note that the assumption of the formulae $\varphi_1, \ldots, \varphi_k$ having a satisfiability problem in NP can be removed [56]. But this assumption is standard in satisfiability modulo theories solvers.

Reading the formula, we see that a bit-string l corresponds to the Venn region $S_1^{l(1)} \cap \ldots \cap S_k^{l(k)}$ of the Venn diagram generated by the sets $S_1, \ldots, S_k$. This fact is important for the decision procedure given in [59] since the procedure has to make sure that the cardinality constraints from QFBAPA have to agree with the cardinality constraints the regular expression imposes on the elementary Venn regions that it uses.

It is nearly inevitable to wonder whether this verbosity is really necessary, i.e. must the user specify each elementary Venn region manually? What if the user wants to refer to a Venn region that is not elementary? Then, the user would need to specify, in the worst case, an exponential number of regions. A different train of thought would lead to the same conclusion. Namely, if we are considering reasoning in the algebra (2), it should be the case that the operators can be applied to any sets, and in particular the operator $\prec$ should be applicable to arbitrary Venn regions.

The reason this succinct representation was not given in Definition 19 is that until now, it was not clear how to compute the Parikh image of the resulting symbolic automaton. The first main contribution of the paper is a method to solve this problem that connects with Feferman-Vaught's theorem by requiring a partitioning argument in the decomposition algorithm (see [22, Theorem 3.1.3]). The novelty of the technique is demonstrated by the fact that it was discovered concurrently by two different teams of researchers in the areas of symbolic automata [56] and string solving [26].

2.5. Computational complexity

We assume basic definitions in the theory of computation [4,62] such as NP-hardness and NP-completeness. We will use the notion of polynomial-time verifier which is equivalent to that of non-deterministic polynomial-time procedure with the difference that the non-deterministic computation is encoded as a *certificate*.

Definition 20. A language $L \subseteq \{0,1\}^*$ is in NP if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time Turing machine V, called the verifier for L such that for every $x \in \{0,1\}^*$, $x \in L$ if and only if there exists $C \in \{0,1\}^{p(|x|)}$ such that $V(x,C) = 1$. If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $V(x,C) = 1$, then C is called a certificate for x.

It is straightforward to generalise the notion of polynomial-time verifier so that it outputs a bit-string rather than a single bit. We use this notion to define NP-reductions which we use to formalise the idea that it suffices to solve formulae of the existential fragment that are conjunctions of literals.

Definition 21. A language $L \subseteq \{0,1\}^*$ is NP-reducible to a language $L' \subseteq \{0,1\}^*$, written $L \leq_{np} L'$, if there is a polynomial-time verifier V such that for every $x \in \{0,1\}^*$, $x \in L$ if and only if there exists a certificate C such that $V(x,C) \in L'$.

Lemma 22. *The relation $\leq_{np}$ satisfies the following properties:*

1. *If $L \leq_{np} L'$ and $L' \in NP$ then $L \in NP$.*
2. *If $L \leq_{np} L'$ and $L' \leq_{np} L''$ then $L \leq_{np} L''$.*

Proof. Ad 1), by hypothesis, we have a verifier V satisfying Definition 21 for $L \leq_{np} L'$. We also have a verifier V' accepting L'. Then $V'' = V' \circ V$ is a verifier for L. Ad 2), if V satisfies Definition 21 for $L \leq_{np} L'$ and V' satisfies Definition 21 for $L' \leq_{np} L''$ then $V' \circ V$ satisfies Definition 21 for $L \leq_{np} L''$. □

The following lemma allows our decision procedures to guess with a disjunct from the disjunctive normal form of the existential formula given as input.

Lemma 23. *Let ψ be a first-order formula in prenex form and C a disjunct of the DNF form of its matrix. Then $|C| = O(|\psi|)$.*

Proof. The DNF conversion only affects the propositional structure of the formula. Thus, in C one may at most have the relations occurring in ψ and their negations. In the worst case, one gets at most $2|\psi|$ symbols accounting for the relations and at most $4|\psi|$ symbols accounting for the conjunctions and negations. Therefore, $|C| \leq 6 \cdot |\psi|$. □

The decision procedure for this problem will start by guessing a disjunct of the DNF form of the input formula. To ensure this can be done by a polynomial-time verifier we need to ensure that the size of that disjunct is polynomial in the size of the input formula.

Lemma 24. *There is NP-reduction from the satisfiable formulae of the existential fragment of a theory to the satisfiable formulae of the existential fragment of the theory in conjunctive form.*

Proof. We define a verifier V that given an input x and a certificate w, interprets the certificate as a disjunct φ of the DNF of x and checks that it is so. To check that the guessed disjunct is part of some DNF the verifier only needs to check the formula $\text{prop}(\varphi) \implies \text{prop}(x)$, where prop is a function that transforms the original formula into propositional one syntactically: two atomic formulae are mapped to the same variable if and only if they coincide syntactically.

V is a polynomial-time verifier. Indeed, if x is satisfiable then there must exist a satisfiable disjunct in its DNF form. By Lemma 23, this disjunct of the DNF form can be written in linear space. Thus, there exists a certificate C of polynomial-size encoding such disjunct. Thus, the output of $V(x,C)$ is a satisfiable formula in conjunctive form. Conversely, if the output of $V(x,C)$ is a satisfiabile formula in conjunctive form, by the check that V performs, it follows that $V(x,C)$ is a disjunct of some DNF form of x. It follows that x itself was also satisfiable. □

3. Succinct representation of the ordering relation

To materialise the succinct ordering discussed in Section 2, we will need to modify Definition 19 so that the regular expression $R(l_1,\ldots,l_s)$ over bit-strings used becomes a regular expression $R(L_1,\ldots,L_s)$ over propositional formulas $L_1,\ldots,L_s$ that use the set variables $S_1,\ldots,S_k$. We call these regular expressions symbolic.

Symbolic regular expressions can be interpreted as denoting two different kinds of sets.

In the first interpretation, we interpret the regular expression as specifying a set of words over the alphabet $\{L_1,\ldots,L_s\}$. We call these words symbolic tables. The set of all these words is denoted by $M_S(L_1,\ldots,L_s)$.

In the second interpretation, we interpret the regular expression as specifying a set of words over the alphabet $\{0,1\}^k$ resulting from substituting in each word in $M_S(L_1,\ldots,L_s)$, the letters forming the word by bit-strings in $\{0,1\}^k$ that satisfy the propositional formula labelling each position of the word. We call these words concrete tables. The set of all these words is denoted by $M(L_1,\ldots,L_s)$.

We say that a concrete table corresponds to a symbolic table if the concrete table can be obtained by substituting bit-strings satisfying the propositional letters of the symbolic table.

As an example of how these succinct regular expressions reduce the verbosity of the specification language, suppose that we wanted to specify that an array is formed by two consecutive values one of which lies in the blue region and the other in the yellow region of Fig. 2. If we wanted to use the regular expressions of [59], we would have to write a regular expression for each elementary Venn region (see Definition 14). With the introduced symbolic regular expressions this can be achieved by writing the regular expression $S_3(S_2 \wedge \neg S_3)$.

We give next the update of Definition 19 using symbolic regular expressions.

Definition 25. Consider conjunctions of formulas F in QFBAPA, regular expressions R over s propositional letters and component structure specifications φ_i with $1 \leq i \leq k$, i.e. of the form

$$\begin{aligned} &\exists S_1, \ldots, S_k. F(S_1, \ldots, S_k) \wedge \\ &\quad \exists t_1, \ldots, t_k. \left((t_1, \ldots, t_k) \in R(L_1, \ldots, L_s) \wedge \bigwedge_{i=1}^{k} S_i = \{n \in \mathbb{N} | t_i(n)\} \right) \wedge \\ &\quad \exists x_1, \ldots, x_m. \bigwedge_{i=1}^{k} S_i = \{n \in \mathbb{N} | \varphi_i(x_1(n), \ldots, x_m(n))\} \end{aligned} \tag{3}$$

where φ_i are formulae of a theory whose satisfiability problem is decidable in NP. We call this theory SymRegExp-QFBAPA-Power.

We consider the problem of giving a decision for this theory. To this end, it will be useful to first consider the case where the propositional formulae $L_1, \ldots, L_s$ of the regular expression R denote disjoint Venn regions. In this case, checking satisfiability of a formula in the fragment reduces to determining whether there exists a symbolic table $\overline{s}$ such that the number of occurrences of the propositional letters $L_1, \ldots, L_s$ matches the cardinalities of their denoted regions. This is because when substituting the propositional letters by bit-strings, there is no restriction on which bit-string to choose other than satisfying the cardinality constraints coming from the QFBAPA sub-formula and the set interpretations with the formulas φ_i. To compute the number of times a certain propositional letter occurs, our decision procedure computes the so-called Parikh image of the regular language $M_S(L_1, \ldots, L_s)$.

Definition 26 *(Parikh image).* The Parikh image of $M_S(L_1, \ldots, L_s)$ is the set

$$\mathsf{Parikh}(M_S(L_1, \ldots, L_s)) = \{(|\overline{s}|_{L_1}, \ldots, |\overline{s}|_{L_s}) | \overline{s} \in M_S(L_1, \ldots, L_s)\}$$

where $|\overline{s}|_{L_i}$ denotes the number of occurrences of the propositional formula L_i in the symbolic table $\overline{s}$.

We will use a description of the Parikh image in terms of linear-size existential Presburger arithmetic formulae first obtained by Seidl, Schwentick, Muscholl and Habermehl.

Lemma 27 *([60]). The set* Parikh$(M_S(L_1, \ldots, L_s))$ *is definable by an existential Presburger formula* ρ *of size* $O(|M|)$ *where* $|M|$ *is the number of symbols used to describe the automaton* M.

In the general case, when propositional letters denote overlapping Venn regions, a partitioning argument is required. This argument allows to avoid having to specify an exponential number of cardinalities of the elementary Venn regions contained in the overlap of several propositional letters. Instead, the partition specifies which indices have been generated by each propositional letter $\{L_1, \ldots, L_s\}$ in the symbolic regular expression $R(L_1, \ldots, L_s)$. The idea is formalised in Theorem 28.[6]

First, we fix some notation. We set $p_\beta := \bigcap_{i=1}^{k} S_i^{\beta_i}$ where $\beta \in \{0,1\}^k$, $p_L := \bigcup_{\beta \vDash L} p_\beta$ where L is a propositional formula and $\vDash$ is the propositional satisfaction relation that is true if and only if the assignment of the values in β to the free variables in L makes the formula L true. When using the interpretation of sets of the form:

$$S_i = \{n \in \mathbb{N} | \varphi_i(x_1(n), \ldots, x_m(n))\}$$

the formula defining the Venn region p_β will be denoted by:

$$\varphi^\beta(d) := \bigwedge_{i=1}^{k} \varphi_i^{\beta(i)}(d)$$

where as usual $\beta(i)$ denotes the i-th bit of β and $\varphi^0 := \neg\varphi$, $\varphi^1 := \varphi$.

We write $S_1 \dot{\cup} S_2$ to denote the set $S_1 \cup S_2$ where we want to emphasise that $S_1 \cap S_2 = \emptyset$. Finally, we use the notation $[n] := \{1, \ldots, n\}$ to refer to the first n natural numbers.

Theorem 28. *Given a formula in SymRegExp-QFBAPA-Power, we may compute in polynomial time an equivalent formula (4) in the combination of the quantifier-free fragment of Boolean algebra with Presburger arithmetic and the existential fragment of the alphabet theory.*

We give next the technical statement of the equivalent formula (4). It has three parts. The first is an existential prefix that is shared between both subtheories

[6] It is noteworthy that such a partitioning argument also appears in [22, Theorem 3.1].

$$\exists p \in [s], \sigma : [p] \hookrightarrow [s], \beta_1, \ldots, \beta_p \in \{0,1\}^k$$

where the symbol $\hookrightarrow$ indicates that σ is an injection from $\{1,\ldots,p\}$ to $\{1,\ldots,s\}$. Recall that s is the number of propositional letters of the regular expression and k is the number of set variables used. Thus, the procedure needs to guess $O(\max\{s,k\}^2)$ bits. This is comparable with Nelson-Oppen's combination procedure [46] which needs to decide which pairs of variables are in an equivalence relation.

Formula (4) then requires solving two sub-formulas. First, one needs to solve a formula in the existential fragment of the theory of the elements of the array:

$$C(\beta_1,\ldots,\beta_p) := \bigwedge_{j=1}^{p} \exists d.\varphi^{\beta_j}(d)$$

The second part corresponds to the remaining sub-term and falls within the quantifier-free first-order theory of Boolean algebra with Presburger arithmetic (QFBAPA) [38]:

$$\begin{aligned} H(k_1,\ldots,k_s,S_1,\ldots,S_k,P_1,\ldots,P_s) :=& F(S_1,\ldots,S_k) \wedge \rho(k_1,\ldots,k_s) \wedge \\ & \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \cup_{i=1}^{s} p_{L_i} = \dot{\cup}_{i=1}^{s} P_i \wedge \\ & \bigwedge_{i=1}^{p} |P_i| = k_{\sigma(i)} \wedge \bigwedge_{i=1}^{s} p_{\beta_i} \cap P_i \neq \emptyset \end{aligned}$$

where ρ is the arithmetic expression in Lemma 27 and $|\cdot|$ denotes the cardinality of the argument set expression. The variables $S_1,\ldots,S_k$ stand for the generators of the algebra, while $P_1,\ldots,P_s$ are set to form a partition of the area covered by the propositional letters.

Finally, formula (4) would be as follows,

$$\begin{aligned} &\exists p \in [s], \sigma : [p] \hookrightarrow [s], \beta_1,\ldots,\beta_p \in \{0,1\}^k. \\ &\quad C(\beta_1,\ldots,\beta_p) \wedge \exists \overline{k}, \overline{S}, \overline{P}. H(\overline{k}, \overline{S}, \overline{P}) \end{aligned} \tag{4}$$

where $\overline{k}, \overline{S}, \overline{P}$ abbreviates the full list of variables of the formula F. It is obvious that this formula can be computed in polynomial time.

We proceed next to the proof of the theorem.

Proof. $\Rightarrow$) If a formula from SymRegExp-QFBAPA-Power is satisfiable, then there are sets $S_1,\ldots,S_k$, a word d and a table $\overline{t}$ satisfying

$$\begin{aligned} &F(S_1,\ldots,S_k) \wedge \bigwedge_{i=1}^{k} S_i = \{n \in \mathbb{N} \mid \varphi_i(d(n))\} \wedge \\ &\overline{t} \in M(L_1,\ldots,L_s) \wedge \bigwedge_{i=1}^{k} S_i = \{n \in \mathbb{N} \mid t_i(n)\} \end{aligned} \tag{5}$$

Let $\overline{s} \in M(L_1,\ldots,L_s)$ be the symbolic table corresponding to $\overline{t}$. We define $k_i := |\overline{s}|_{L_i}, p = |\{i \mid k_i \neq 0\}|$, σ mapping the indices in $[p]$ to the indices of the terms for which k_i is non-zero and $P_i = \{n \in \mathbb{N} \mid \overline{s}(n) = L_{\sigma(i)}\}$. It will be convenient to work out the following equalities, which follow from (5):

$$\begin{aligned} p_{L_i} &= \bigcup_{\beta \vDash L_i} \bigcap_{j=1}^{k} S_j^{\beta_j} = \bigcup_{\beta \vDash L_i} \{n \in \mathbb{N} \mid \bigwedge_{j=1}^{k} t_j^{\beta_j}(n)\} = \{n \in \mathbb{N} \mid \overline{t}(n) \vDash L_i\} \\ p_{L_i} &= \bigcup_{\beta \vDash L_i} \bigcap_{j=1}^{k} S_j^{\beta_j} = \bigcup_{\beta \vDash L_i} \{d \in \mathcal{D} \mid \bigwedge_{j=1}^{k} \varphi_j^{\beta_j}(d)\} = \{d \in \mathcal{D} \mid L_i(\overline{\phi}(d))\} \end{aligned} \tag{6}$$

where $L_i(\overline{\phi}(d(n)))$ is the propositional formula obtained by substituting set variables by the formulae $\phi_i(d(n))$. We now deduce formula (4):

- $F(S_1,\ldots,S_k)$ holds by assumption.
- $\rho(k_1,\ldots,k_s)$: from $\overline{s} \in P(L_1,\ldots,L_s)$, we have that

$$(k_1,\ldots,k_s) \in \mathsf{Parikh}(M_S(L_1,\ldots,L_m))$$

and therefore $\rho(k_1,\ldots,k_s)$.

- $P_i \subseteq p_{L_{\sigma(i)}}$: since $\overline{s}$ corresponds to $\overline{t}$, for all $n \in \mathbb{N}$ we have $\overline{t}(n) \vDash \overline{s}(n)$ and the inclusion follows from the definition of P_i and equality (6).
- $|P_i| = k_{\sigma(i)}$: since $|P_i| = \left|\{n \in \mathbb{N} | \overline{s}(n) = L_{\sigma(i)}\}\right| = |\overline{s}|_{L_{\sigma(i)}} = k_{\sigma(i)}$.
- Each pair of sets P_i, P_j with $i < j$ is disjoint:

$$P_i \cap P_j = \{n \in \mathbb{N} | \overline{s}(n) = L_{\sigma(i)}\} \cap \{n \in \mathbb{N} | \overline{s}(n) = L_{\sigma(j)}\} =$$
$$= \{n \in \mathbb{N} | \overline{s}(n) = L_{\sigma(i)} = L_{\sigma(j)}\} = \emptyset$$

  using that the letters L are chosen to be distinct and that σ is an injection (so $\sigma(i) \neq \sigma(j)$).
- $p_{L_1} \cup \ldots \cup p_{L_s} = P_1 \dot{\cup} \ldots \dot{\cup} P_p$: since by definition $P_i = \{n \in \mathbb{N} | \overline{s}(n) = L_{\sigma(i)}\}$, $p_{L_i} = \{n \in \mathbb{N} | \overline{t}(n) \vDash L_i\}$ and by definition of σ it follows that the only letters that can appear in $\overline{s}$ are $L_{\sigma(1)}, \ldots, L_{\sigma(p)}$. Thus, we have $p_{L_1} \cup \ldots \cup p_{L_s} = [1, |\overline{t}|] = [1, |\overline{s}|] = P_1 \dot{\cup} \ldots \dot{\cup} P_p$.
- There exists $\beta_1, \ldots, \beta_p \in \{0,1\}^k$, such that $\bigwedge_{i=1}^{s} P_{\beta_i} \cap P_i \neq \emptyset$: note that $P_i \neq \emptyset$ by definition of σ. Thus, there must exist some β_i such that $p_{\beta_i} \cap P_i \neq \emptyset$. We pick any such β_i.
- $\bigwedge_{j=1}^{p} \exists d.\varphi^{\beta_j}(d)$: follows from $p_{\beta_j} \cap P_j \neq \emptyset$ and formula (6).

$\Leftarrow$) Conversely, if formula (4) is satisfiable, then there is an integer $p \in [s]$, an injection $\sigma : [p] \hookrightarrow [s]$, bit-strings $\beta_1, \ldots, \beta_p \in \{0,1\}^k$, integers $k_1, \ldots, k_s$ and sets $S_1, \ldots, S_k, P_1, \ldots, P_p$ satisfying

$$\bigwedge_{j=1}^{p} \exists d.\varphi^{\beta_j}(d) \wedge$$
$$F(S_1, \ldots, S_k) \wedge \rho(k_1, \ldots, k_s) \wedge \bigwedge_{i=1}^{p} P_i \subseteq p_{L_{\sigma(i)}} \wedge \cup_{i=1}^{s} p_{L_i} = \dot{\cup}_{i=1}^{p} P_i \wedge \tag{7}$$
$$\bigwedge_{i=1}^{p} |P_i| = k_{\sigma(i)} \wedge \bigwedge_{i=1}^{p} p_{\beta_i} \cap P_i \neq \emptyset$$

From $\rho(k_1, \ldots, k_s)$ follows that there is a symbolic table $\overline{s} \in M_S(L_1, \ldots, L_s)$ such that $|\overline{s}|_{L_i} = k_i$ for each $L_i \in \{L_1, \ldots, L_s\}$. From

$$p_{L_1} \cup \ldots \cup p_{L_s} = P_1 \dot{\cup} \ldots \dot{\cup} P_p \wedge \bigwedge_{i=1}^{p} P_i \subseteq p_{L_{\sigma(i)}} \wedge \bigwedge_{i=1}^{p} |P_i| = k_{\sigma(i)}$$

follows that we can replace the formulae L_i occurring in the symbolic table $\overline{s}$ by the bit-strings representing the elementary Venn regions to which the indices of the sets P_i belong. Moreover, thanks to the condition $\bigwedge_{i=1}^{p} p_{\beta_i} \cap P_i \neq \emptyset$ follows that we can replace the letters L_i by the bit-strings β_i, defining $\overline{t}$ as $\overline{t}(n) = \begin{cases} \beta_i & \text{if } n \in P_i \end{cases}$. In this way, we obtain a table $\overline{t} \in M(L_1, \ldots, L_s)$. We then define the corresponding word over $\mathcal{D}$, thanks to the property $\bigwedge_{i=1}^{p} \exists d.\phi^{\beta_i}(d)$. Naming the witnesses of these formulae as d_i, we define $d(n) = \begin{cases} d_i & \text{if } n \in P_i \end{cases}$. To conclude, note that:

$$\{n \in \mathbb{N} | t_j(n)\} = \cup_{\{1 \leq i \leq k | \beta_i(j) = 1\}} P_i = \{n \in \mathbb{N} | \varphi_j(d(n))\}$$

Thus, we have that the formula of Definition (25) is satisfied by the set variables

$$S_j := \{n \in \mathbb{N} | t_j(n)\} = \{n \in \mathbb{N} | \varphi_j(d(n))\} \quad \square$$

We note that the reduction procedure that we give comes with a computational complexity guarantee.

Corollary 29. *Deciding formulas in SymRegExp-QFBAPA-Power is a NP-complete problem.*

Proof. The problem is trivially NP-hard. A non-deterministic polynomial time algorithm would simply guess the values

$$\exists p \in [s].\sigma : [p] \hookrightarrow [s].\exists \beta_1, \ldots, \beta_p \in \{0,1\}^k$$

and use existing routines for QFBAPA and the existential theory of the theory of the elements of the array. $\square$

4. The case of trees

This section shows how the decision procedure for checking satisfiability formulated in the previous section, can be adapted to verify tree data structures. Again the key issue would be computing the Parikh image of the symbolic regular tree language that arises. We draw the connection here since it relies on different results, namely [33,73] and the connection is not necessarily immediate.

The notion of tree can be given as follows (cfr. [18]):

Definition 30. A binary Σ-tree is a function $\tau : A \to \Sigma$ where A is a finite subset of $\{0,1\}^*$ closed under the initial segment relation (i.e. if $uv \in A$ then $u \in A$). $\Sigma^\#$ is the class of all binary Σ-trees. Λ is the function with domain $\emptyset$ also known as the empty tree. For $\sigma \in \Sigma$ and $\tau, \tau' \in \Sigma^\#$, $\sigma[\tau, \tau']$ is the Σ-tree with a root σ, left subtree τ and right subtree τ'.

Doner [18] seems to have been the first to recognise the connection between the weak monadic second-order theory of two successors, i.e. the theory of the structure $\langle \{0,1\}^*, x \mapsto x0, x \mapsto x1 \rangle$, and recognisability by tree automata. The connection is similar to that of Büchi, which we described in the preliminaries section. We start with the definition of tree automaton.

Definition 31. A tree automaton M over an alphabet Σ is a tuple (Q, q_0, F, Δ) where Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states and $\Delta \subseteq Q \times \Sigma \times Q \times Q$ is a finite set of transitions.

The language of M at state $q \in Q$, denoted by $\mathcal{L}_q(M)$, is the smallest subset of $\Sigma^\#$ such that if $q \in F$ then $\Lambda \in \mathcal{L}_q(M)$, if $(q_1, a, q_2, q_3) \in \Delta$, and for $i \in \{1,2\}$, $\tau_i \in \mathcal{L}_{q_i}(M)$, then $a[\tau_1, \tau_2] \in \mathcal{L}_{q_3}(M)$. The language of M is $\mathcal{L}(M) = \mathcal{L}_{q_0}(M)$.

Doner's theorem says that if we take the set of trees accepted by a tree automaton over bit-string letters, i.e. with $\Sigma = \{0,1\}^k$ and we compute for each position of the bit-string the positions in the tree that have a 1-entry for that position, then the resulting sets are definable in the weak monadic second-order theory of the structure $\langle \{0,1\}^*, x \mapsto x0, x \mapsto x1 \rangle$.

Again, we could consider an automaton that specifies the letters using propositional formulas rather than specific bit-strings. This is done, for instance, in the setting of symbolic tree automata [32,24,71,15,14] and following Definition 25 we could express the language of such a symbolic tree automata (with cardinalities) as in the following definition.

Definition 32. Consider conjunctions of formulas F in QFBAPA, regular tree expressions R over s propositional letters and component structure specifications φ_i with $1 \le i \le k$, i.e. of the form

$$\begin{aligned}
&\exists S_1, \ldots, S_k. F(S_1, \ldots, S_k) \wedge \\
&\quad \exists \tau. \left(\tau \in R(L_1, \ldots, L_s) \wedge \bigwedge_{i=1}^{k} S_i = \{n \in \{0,1\}^* \mid \tau_i(n)\} \right) \wedge \\
&\quad \exists \overline{x}. \bigwedge_{i=1}^{k} S_i = \{n \in \{0,1\}^* \mid \varphi_i(\overline{x}(n))\}
\end{aligned}$$

where φ_i are formulae of a theory whose satisfiability problem is decidable in NP. We call this theory SymTreeExp-QFBAPA-Power.

As announced in the introduction, for checking satisfiability or, equivalently, non-emptiness of the tree automaton, we need to compute the Parikh image of the symbolic regular tree language that arises. The definition of Parikh image is analogous to that of Definition 26.

Definition 33 *(Parikh Image).* The Parikh image of $M_S(L_1, \ldots, L_s)$ is the set

$$\mathsf{Parikh}(M_S(L_1, \ldots, L_s)) = \{(|s|_{L_1}, \ldots, |s|_{L_s}) \mid \overline{s} \in M_S(L_1, \ldots, L_s)\}$$

where $|s|_{L_i}$ denotes the number of occurrences of the propositional formula L_i in the symbolic tree table $\overline{s}$.

In this setting, we need to adapt the result of [60]. In particular, to compute the Parikh image of a regular tree language, we use a key observation by Klaedtke and Rueß [33] that allows to reduce the problem to that of computing the Parikh image of a context-free grammar.

First, note that the tree language $M_S(L_1, \ldots, L_s)$ is given by a non-deterministic bottom-up tree automaton (this follows from Definition 31). However, the construction of Klaedtke and Rueß is given by non-deterministic top-down tree automata.

Definition 34. A non-deterministic top-down tree automaton M over an alphabet Σ is a tuple $\big(Q, q_0, F, \Delta\big)$, where Q is a finite set of states, q_0 is the initial state, $F \subseteq Q$ is the set of final states, and $\Delta : Q \times \Sigma \to \mathcal{P}(Q \times Q)$ is the transition function. Associated with M is the function $\overline{\Delta} : \Sigma^\# \to S$ defined by $\overline{\Delta}(\Lambda) = \Lambda$ and $\overline{\Delta}(\sigma[\tau, \tau']) = \Delta(q_0, \sigma)$.

A run ρ of M on $t \in \Sigma^\#$ is a Q-labelled tree ρ with $\mathrm{dom}(\rho) = \{\epsilon\} \cup \{ub \mid u \in \mathrm{dom}(t), b \in \{0,1\}\}$ such that $\rho(u) = q_0$, and $(\rho(u0), \rho(u1)) \in \Delta(\rho(u), t(u))$, for all $u \in \mathrm{dom}(t)$. ρ is accepting if all leaves of ρ are labelled with states in F, i.e. $\rho(u) \in F$, for all $u \in \mathrm{dom}(\rho) \setminus \mathrm{dom}(t)$. A tree t is recognised by M if there is an accepting run of M on t. $T(M)$ denotes the set of trees that are recognised by M.

Fortunately, it is easy to convert from non-deterministic top-down to non-deterministic bottom-up tree automata.

Proposition 35 *([12, Theorem 1.6.1]). The class of languages accepted by top-down NFTAs is precisely the class of languages accepted by bottom-up NFTAs. Given a top-down (bottom-up) NFTA one can compute a bottom-up (top-down) NFTA in linear time in the number of edges and states of the input.*

Second, we use the observation of Klaedtke and Rueß [33, Lemma 17] to compute a context-free grammar with the same Parikh image.

Lemma 36 *([33]). For any non-deterministic top-down tree automaton $\mathcal{A}$ one can compute in linear time a context-free grammar $G_{\mathcal{A}}$ expressing the trees accepted by $\mathcal{A}$ as words obtained through the in-order traversal of the trees. As a consequence $\mathsf{Parikh}(\mathcal{A}) = \mathsf{Parikh}(G_{\mathcal{A}})$.*

Proof. Let $\mathcal{A} = \left(Q, \Gamma, \delta, q_I, F\right)$ be a top-down tree automaton. We define a context-free grammar $G = \langle V, \Sigma, R, S\rangle$ that generates the words obtained by traversing the trees recognised by $\mathcal{A}$ in infix order as follows:

- $V = Q$ is the set of non-terminal symbols.
- Σ is the set of terminal symbols.
- There are two kinds of derivation rules:
  - For each $\left(q, q'\right) \in \delta(p, b)$, we have the rule $p \to qbq'$.
  - If $(F \times F) \cap \delta(q, b) \neq \emptyset$ then we have the rule $q \to b$.
- $S = q_I$ is the start symbol of G.

It is immediate from the definition that $Inorder(L(\mathcal{A})) = L(G)$ and that the size of the grammar is equal to that of the automaton. Since the Parikh image is invariant under permutation of the labels, it follows that $\mathsf{Parikh}(\mathcal{A}) = \mathsf{Parikh}(G_{\mathcal{A}})$. □

The third key observation, by Verma, Seidl and Schwentick, is that the Parikh image of a context-free grammar can be described by a linear-sized existential Presburger arithmetic formula.

Lemma 37 *([73]). Given a context-free grammar G, one can compute an existential Presburger formula ϕ_G for the Parikh image of $L(G)$ in linear time.*

In summary, we have the following auxiliary result.

Lemma 38. *The set $\mathsf{Parikh}(M_S(L_1, \ldots, L_n))$ is definable by an existential Presburger formula ρ of size $O(|M|)$ where $|M|$ is the number of symbols used to describe the automaton M.*

With these adjustments, the proof proceeds as in Section 3.

Corollary 39. *Given a formula in SymTreeExp-QFBAPA-Power, we may compute in polynomial time an equivalent formula in the combination of the quantifier-free fragment of Boolean algebra with Presburger arithmetic and the existential fragment of the alphabet theory.*

Corollary 40. *Deciding formulas in SymTreeExp-QFBAPA-Power is a NP-complete problem.*

5. The need for aggregators

Most constraint languages in use in databases [28] or verification [13] use some form of aggregation operators. These are useful in applications that often deal with counting, adding quantities, computing maximums, or minimums. One recent area of application for these languages is the verification of smart contracts [51,1,21,3,47] where one often needs to compute these quantities.

We have already discussed the cardinality aggregator. In the following sections, we extend the results of [54] and [37] to handle these aggregation operators over arrays. Our proofs work by reduction to the language QFBAPAI [59] (or equivalently the combination QFBAPA-Power) and on each occasion we manage to prove that the resulting satisfiability problem is NP-complete. These results lift the restriction mentioned in the original combinatory array logic paper [16] that the cardinality of multisets could not be encoded in the fragment. Since sums of elements in the array correspond to cardinalities of the arrays, this restriction no longer applies. Summation constraints were also absent of the paper that introduced array folds logic [13]. It appears that summation constraints were of interest to the authors of [13] since the repository of the tool AFOLDER [50] contains some (commented out) examples using sums. However, array fold logic as presented in [13] cannot express sums, since counters can only be updated by adding constants.

6. Minimum and maximum aggregators

When we discuss minimum and maximum, we need to distinguish whether these are taken with respect to the indices of the arrays or their contents. For simplicity, we assume that we work with arrays indexed by natural numbers and taking values over the natural numbers.

6.1. Elimination of the minimum and maximum aggregators over index sets

In the case of the indices, this construction was already discussed in [37], essentially, one needs to deal with terms $\inf(S)$ and $\sup(S)$ and introduces integer variables for them.

One then guesses which of these inf and sup are infinite and which finite (see [37, Guessing the empty and the infinite sets]). For instance, if we guess that the term $k = \max(S)$ is infinite, then we introduce the constraint $k \geq \aleph_0$.

For the remaining finite variables, one guesses an ordering between the variables and introduces constraints between the classes of variables that are guessed to be equal. For instance, if $k_1 = k_2 = k_3$ are guessed to be less than $k_4 = k_5$ then one introduces variables b_1, b_2 to represent the classes and then imposes the constraint $b_1 < b_2$ (see [37, Guessing an ordering on bounds]).

We already showed in [59, Section 6] that reasoning in QFBAPAI is compatible with infinite constraints in the extension $\mathsf{QFBAPA}_{\aleph_0}$. It follows that the elimination of the maximum and minimum operators can also be carried out in QFBAPAI.

6.2. Elimination of the minimum and maximum aggregators over elements

Unlike in the case of indices, the maximum and the minimum value of the elements of an array always exists; thus, we are entitled to use the notation min and max in what follows.

It turns out that the maximum of an array x, $\max(x)$ or its minimum, $\min(x)$, are easily eliminated in our language by introducing integer variables replacing their occurrences in the set interpretations. More precisely, if we have a set of interpretations of the form:

$$S = \{n \in \mathbb{N} | \varphi_i(\overline{x}(n), \max(x_j))\}$$

then we replace it by

$$\exists k.S = \{n \in \mathbb{N} | \varphi_i(\overline{x}(n), k)\}$$

noting that the language of QFBAPAI as introduced in [58] allows such quantified integer constants to appear in set interpretations even if in posterior developments we drop them for convenience.

Now, to obtain an equivalent formula, we need to make sure that k is, in fact, the maximum of the array x_j. For this, we introduce two auxiliary set interpretations

$$S_1 = \{n \in \mathbb{N} | x_j(n) \leq k\} \wedge S_2 = \{n \in \mathbb{N} | x_j(n) = k\}$$

and introduce the QFBAPA constraints

$$\overline{S_1} = \emptyset \wedge |S_2| \geq 1$$

which ensure, respectively, that all the elements of the array x_j are smaller or equal than k and that there exists at least one equal to k.

The case for the minimum operator is completely analogous.

7. Summation aggregator

Our decision procedure works by reduction to Presburger arithmetic and the theory of the elements of the array. We will do so by successively eliminating the component theories, following the composition method of Feferman and Vaught. Our first step is thus defining the input language to be transformed into these constraints.

Definition 41. The theory QFBAPA-Power-Sum consists of formulae of the form

$$F(S_1, \ldots, S_k, \overline{\sigma}) \wedge \bigwedge_{i=1}^{k} S_i = \{i \in I | \varphi_i(\overline{c}(i))\} \wedge \overline{\sigma} = \sum (|\overline{c}(i)| \varphi_0(\overline{c}(i))|) \tag{8}$$

where F is a formula from QFBAPA, $\varphi_0, \ldots, \varphi_k$ are formulae in the existential fragment of Presburger arithmetic and $\overline{c}$ is a tuple of arrays of natural numbers. We will refer to the first conjunct of this formula as the QFBAPA term, to the second conjunct as the set interpretations and to the third conjunct as the multiset interpretations.

There are important differences between Definition 41 and [52, Definition 2.1]. [52, Definition 2.1] has a quantifier-free Presburger arithmetic formula instead of the QFBAPA term F. Second, the term $\forall e.F$ corresponds to the set interpretations. Third, the term $(u_1, \ldots, u_n) = \sum_{e \in \mathbb{E}} (t_1, \ldots, t_n)$ corresponds to our multiset interpretation. It should be noted that the indices in our setting range over the natural numbers and not over a finite set $\mathbb{E}$ as in [52].

An important observation is that the definition does not allow free variables to be shared between the three conjuncts. In fact, if we allowed such shared constants, the resulting fragment would have an undecidable satisfiability problem.

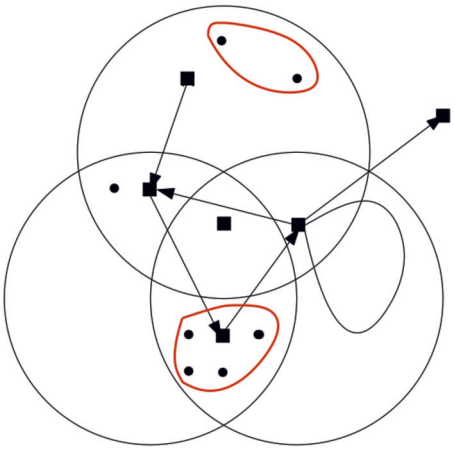


Fig. 4. An example Venn diagram with regular and sum constraints.

Corollary 42. *The satisfiability of formulas of the form*

$$F(S_1,\ldots,S_k,\overline{\sigma},\overline{f}) \wedge \bigwedge_{i=1}^{k} S_i = \{i \in I \mid \varphi_i(\overline{c}(i),\overline{f})\} \wedge \overline{\sigma} = \sum(|\overline{c}(i)| \varphi_0(\overline{c}(i),\overline{f})|) \tag{9}$$

is undecidable.

Proof. By reduction from Hilbert's tenth problem [42]. One can encode in this theory the addition of two natural numbers using the formula F which is in QFBAPA and thus includes quantifier-free Presburger arithmetic. Multiplication $z = xy$ can be encoded by imposing the array $\overline{c}$ to be equal to the constant x in each position, have length y and sum up to z. □

The resulting constraints are schematically represented in Fig. 4. The red curves encircling some dots represent those indices whose contents are contributing to the sum $\overline{\sigma}$. As per the formulation of Definition 41, these are exactly those contents that satisfy the formula φ_0. In rebuilding the model, we will need to distinguish those elements that do not satisfy φ_0 which can be freely replicated, from those that satisfy φ_0 which will need a more sophisticated check.

7.1. Elimination of the Boolean algebra with Presburger arithmetic terms

With the goal of eliminating the QFBAPA constraints, we introduce k array variables $c_1,\ldots,c_k$ and we rewrite the Boolean algebra expressions and cardinality constraints in terms of set interpretations and summation constraints. As follows.

- For every newly introduced array variable c_j, we introduce the set interpretation:

  $$I = \{i \in I \mid c_j(i) = 0 \vee c_j(i) = 1\}$$

  this constraint is added to the set interpretation term.
- For every newly introduced array variable c_j, we rewrite the set variable S_j into the following set interpretation:

  $$S_j := \{i \in I \mid c_j(i) = 1\}$$

  which says that the indices in S_j correspond to the positions where c_j is equal to one.
- We then substitute each Boolean algebra expression appearing in the QFBAPA term of Formula (8), by repeatedly applying the following rewrite rules:

  $$S_j^c := \{i \in I \mid c_j(i) = 0\}$$
  $$S_j \cup S_k := \{i \in I \mid c_j(i) = 1 \vee c_k(i) = 1\}$$
  $$S_j \cap S_k := \{i \in I \mid c_j(i) = 1 \wedge c_k(i) = 1\}$$

- By the above rewriting process, each cardinality constraint $|S| = k$ is rewritten as

  $$|\{i \in I \mid \varphi(\overline{c}(i))\}| = k$$

where the variable $\overline{c}$ lists the newly introduced array variables $c_1, \ldots, c_k$. We then introduce a new array variable $x = ite(\varphi(\overline{c}), 1, 0)$, that is, $x(i)$ is equal to one if $\varphi(\overline{c}(i))$ holds and it is equal to 0 otherwise. This can be encoded with the set interpretation

$$I = \{i \in I \,|\, x(i) = ite(\varphi(\overline{c}(i)), 1, 0)\}$$

One then rewrites the expression $|S| = k$ into $k = \sum_{i \in I} x(i)$.[7]

As a result of this phase, we obtain a formula of the form:

$$\psi(\overline{\sigma}) \wedge \bigwedge_{i=1}^{k} I = \{i \in I \,|\, \varphi_i(\overline{c}(i))\} \wedge \overline{\sigma} = \sum (|\overline{c}(i) \,|\, \varphi_0(\overline{c}(i))|) \tag{10}$$

where ψ is a quantifier-free Presburger arithmetic formula and all the Boolean algebra and cardinality constraints has been translated into set interpretations and summation constraints.

7.2. Elimination of the set interpretations

The next step in the decision procedure is to eliminate the set interpretation term. However, in the form of Formula (10), this is particularly simple. Formula (10) is equivalent to:

$$\psi(\overline{\sigma}) \wedge \overline{\sigma} = \sum (|\overline{c}(i)| \bigwedge_{i=0}^{k} \varphi_i(\overline{c}(i))|) \tag{11}$$

It thus remains to remove the summation operator.

7.3. Elimination of the summation operator

The next step is to rewrite sums to a star operator introduced in [54]. Given a set A, the set A^* is defined as:

$$A^* = \{u \mid \exists N \geq 0, x_1, \ldots, x_N \in A. u = \sum_{i=1}^{N} x_i\}$$

Proposition 43 *(Multiset elimination). The formula*

$$\exists \overline{\sigma}, \overline{c}. \psi(\overline{\sigma}) \wedge \overline{\sigma} = \sum_{n \in \mathbb{N}} (|\overline{c}(n) \,|\, \varphi(\overline{c}(n))|) \tag{12}$$

and the formula

$$\exists \overline{\sigma}. \psi(\overline{\sigma}) \wedge \overline{\sigma} \in \{\overline{k} \mid \varphi(\overline{k})\}^* \tag{13}$$

are equivalent.

Proof. The argument needs to be adapted from Theorem 2.4 of [55] since both our index and element set are infinite.
$\Rightarrow$) If (1) is satisfied, there are $\overline{\sigma}, \overline{c}$ such that $\psi(\overline{\sigma}) \wedge \overline{\sigma} = \sum_{n \in \mathbb{N}} (|\overline{c}(n) \,|\, \varphi(\overline{c}(n))|)$. We claim that the same $\overline{\sigma}$ satisfies $\psi(\overline{\sigma}) \wedge \bigwedge_{i=1}^{k} \overline{\sigma} \in \{\overline{k} \,|\, \varphi(\overline{k})\}^*$. By hypothesis, $\psi(\overline{\sigma})$ is true. Moreover, $\overline{\sigma} = (|\overline{c}(n) \,|\, \varphi(\overline{c}(n))|)$ and either

- $(|\overline{c}(n) \,|\, \varphi(\overline{c}(n))|)$ is finite in which case $\overline{\sigma} \in \{\overline{k} \,|\, \varphi(\overline{k})\}^*$.
- or $(|\overline{c}(n) \,|\, \varphi(\overline{c}(n))|)$ is infinite, in which case $\overline{c}(n)$ is equal to $\overline{0}$ in all but a finite set of indices I since by hypothesis the sum $\overline{\sigma}$ is finite. Then $\overline{\sigma} = \sum_{n \in I} \overline{c}(n)$ and $\overline{\sigma} \in \{\overline{k} \,|\, \varphi(\overline{k})\}^*$.

$\Leftarrow$) If (2) is satisfied, then there is $\overline{\sigma}$ such that $\psi(\overline{\sigma}) \wedge \overline{\sigma} \in \{\overline{k} \,|\, \varphi(\overline{k})\}^*$. It follows that there is a finite list of elements $\overline{k}_i$ such that $\overline{\sigma} = \sum_{i=1}^{p} \overline{k}_i$. We define

$$\overline{c}(n) = \begin{cases} \overline{k}_n & \text{if } 1 \leq n \leq p \\ 0 & \text{otherwise} \end{cases}$$

It is immediate that $\overline{\sigma}$ and $\overline{c}$ satisfy $\psi(\overline{\sigma}) \wedge \overline{\sigma} = \sum_{n \in \mathbb{N}} (|\overline{c}(n) \,|\, \varphi(\overline{c}(n))|)$. □

[7] It is remarkable than being closed under the if-then-else operator (ite) also appears in modern presentations of Feferman-Vaught theorem, see for instance [29, Theorem 9.6.2], as the property of "being closed under gluing over a Boolean algebra".

On input $\langle x, w\rangle$:

1. Interpret w as:
   (a) lists of vectors a_l, b_s generating some element in each semilinear set given by F_i.
   (b) matrices A_i and vectors b_i such that $A_i x \leq b_i$ is in the disjunctive normal form of each F_i.
   (c) a certificate C for V_{PA} on input y.

   where

$$y = \left(\begin{array}{c|c} \begin{matrix} A_{g_1} \\ \cdots \\ A_{g_n} \end{matrix} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \hline 0 & A_0 \end{array}\right) \left(\begin{matrix} \cdots \\ \lambda_{ij} \\ \cdots \\ \cdots \\ \mu_i \\ \cdots \\ x \\ u \\ v \end{matrix}\right) = \left(\begin{matrix} b_{g_1} \\ \cdots \\ b_{g_n} \\ b_0 \end{matrix}\right) \wedge \bigwedge_{il} A_i a_l \leq b_i \wedge \bigwedge_{is} A_i b_s \leq 0$$

2. Accept iff V_{PA} accepts $\langle y, C\rangle$.

Fig. 5. Verifier for LIA^{card}.

The next step is to eliminate the star operator introduced in Proposition 43. To do so, one could use [55, Theorem 2.23] which shows that if Formula (13) is satisfiable then it also has a solution that can be written with a polynomial number of bits. We adapt this result to the case where we consider explicit integer exponents in the sets. That is we consider given a set A and an integer $m \in \mathbb{N}$, the set A^m defined as:

$$A^m = \{u \mid x_1, \ldots, x_m \in A. u = \sum_{i=1}^{m} x_i\}$$

the reason to do this is that when mixing constraints of the form given in Definitions 25 and 41, it comes it handy to *synchronise* the cardinality constraints coming from the regular expressions and summation constraints and we cannot do that unless we mention explicitly the number of addends used in the sums. An sketch of the proof for this extension follows.

Definition 44. LIA with sum cardinalities, denoted LIA^{card}, is the theory consisting of formulas of the form $F_0 \wedge \bigwedge_{i=1}^{n} u \in \{x \mid F_i(x)\}^{x_i}$ where F_0 and F are quantifier-free Presburger arithmetic formulae.

Proposition 45. *LIA^{card} is in NP.*

Proof. Let V_{PA} be a polynomial time verifier for LIA^{card}. Fig. 5 gives a verifier V for LIA^{card}. We show that $x \in \mathsf{LIA}^{card}$ if and only if there exists a polynomial-size certificate w such that V accepts $\langle x, w\rangle$.

$\Rightarrow$) If $x = F_0 \wedge \bigwedge_{i=1}^{n} u \in \{x \mid F_i(x)\}^{x_i} \in \mathsf{LIA}^{card}$ then we show that there is a solution that uses a polynomial number of bits in the size of F_0 and F_1.

We convert each formula F_i to semilinear normal form [55, Theorem 2.13].

Lemma 46. *Let F be a linear arithmetic formula of size s. Then there exist numbers $m, q_1, \ldots, q_m \in \mathbb{N}$ and vectors $a_i, b_{ij} \in \mathbb{N}^n$ for $1 \leq j \leq q_i, 1 \leq i \leq m$ with $\|a_i\|_1, \|b_{ij}\|_1 \leq 2^{p(s)}$ with p polynomial such that $F(x)$ is equivalent to the formula:*

$$\exists \alpha_{11}, \ldots, \alpha_{mq_m}. \bigvee_{i=1}^{m} \left(x = a_i + \sum_{j=1}^{q_i} \alpha_{ij} b_{ij}\right) \tag{14}$$

Next, we eliminate the star operator, as in [41, Proposition 2] and [55, Theorem 2.14].

Lemma 47. *Let F be a quantifier-free linear integer arithmetic formula whose semilinear normal form is formula (14). Then $u \in \{y \mid F(y)\}^x$ is equivalent to*

$$\exists \overline{\mu}, \overline{\lambda}. u = \sum_{i=1}^{q} \left(\mu_i a_i + \sum_{j=1}^{q_i} \lambda_{ij} b_{ij}\right) \wedge \bigwedge_{i=1}^{q} \left(\mu_i = 0 \implies \sum_{j=1}^{q_i} \lambda_{ij} = 0\right) \wedge x = \sum_{i=1}^{q} \mu_i$$

We express the resulting vector u with polynomially many generators [55, Theorem 2.20].

Lemma 48 *(Polynomially many generators for sums). Let F be a quantifier-free linear integer arithmetic formula of size s whose semilinear normal form is formula (14). Then $u \in \{y | F(y)\}^x$ is equisatisfiable with*

$$\exists \lambda_{ij}, \mu_i . u = \sum_{i \in I_0} \left(a_i + \sum_{(i,j) \in J} \lambda_{ij} b_{ij} \right) + \sum_{i \in I_1} \mu_i a_i \wedge x = |I_0| + \sum_{i \in I_1} \mu_i$$

for some $I_0, I_1 \subseteq \{1, \ldots, q\}$, $J \subseteq \cup_{i=1}^{q} \{(i,1), \ldots, (i,q_i)\}$, $|I_0| \leq |J| \leq q(s)$, $|I_1| \leq q(s)$ and q is a polynomial.

We write the equation

$$u = \sum_{i \in I_0} \left(a_i + \sum_{(i,j) \in J} \lambda_{ij} b_{ij} \right) + \sum_{i \in I_1} \mu_i a_i \wedge x = |I_0| + \sum_{i \in I_1} \mu_i$$

as a system $A_g x_g = b_g$ of the form

$$\begin{pmatrix} \ldots & a_i & \ldots & \ldots & b_{ij} & \ldots & \ldots & a_i & \ldots & 0 & -I_k \\ 0 & \ldots & \ldots & \ldots & \ldots & 0 & 1 & \ldots & 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} \ldots \\ \lambda_{ij} \\ \ldots \\ \ldots \\ \mu_i \\ \ldots \\ x \\ u \end{pmatrix} = \begin{pmatrix} \ldots \\ -a_i \\ \ldots \\ 0 \\ \ldots \\ 0 \\ -|I_0| \end{pmatrix}$$

Since u is also a solution of F_0 it will further satisfy some system $A_0 w = b_0$ corresponding to one of the terms of the disjunctive normal form of F_0 and where we can assume that the unknown u appears in the first rows of $w = (u, v)$. As a result, we obtain a combined system.

$$\left(\begin{array}{c|c} A_{g_1} & 0 \\ \ldots & \ldots \\ A_{g_n} & 0 \\ \hline 0 & A_0 \end{array} \right) \begin{pmatrix} \ldots \\ \lambda_{ij} \\ \ldots \\ \ldots \\ \mu_i \\ \ldots \\ x \\ u \\ v \end{pmatrix} = \begin{pmatrix} b_{g_1} \\ \ldots \\ b_{g_n} \\ b_0 \end{pmatrix}$$

This system has a polynomial number of rows, columns and uses polynomially many bits for its maximum absolute value. Thus, it is guaranteed to have a solution that uses polynomially many bits by the following theorem from [48, page 767].

Lemma 49. *Let A be an $m \times n$ integer matrix and b a m-vector, both with entries from $[-a..a]$. Then the system $Ax = b$ has a solution in $\mathbb{N}^n$ if and only if it has a solution in $[0..M]^n$ where $M = n(ma)^{2m+1}$.*

Moreover, since all the generators chosen for F_i lie in the same linear subset they satisfy $A_i a_l \leq b_i$ and $A_i b_s \leq 0$ for some system $A_i x \leq b_i$ in the disjunctive normal form of each F_i.

The resulting formula has polynomial-size in the size of F_0 and F_1, thus there exists a polynomial-size certificate C such that V_{PA} accepts $\langle y, C \rangle$. It follows that $V_{\mathsf{LIA}^{card}}$ accepts $\langle x, \langle \{a_l\}, \{b_s\}, \{A_i\}, \{b_i\}, C \rangle \rangle$.

$\Leftarrow$) If $V_{\mathsf{LIA}^{card}}$ accepts $\langle x, w \rangle$ then there exists a polynomial-size certificate C such that V_{PA} accepts $\langle y, C \rangle$ and thus y is satisfiable. This means that for some vectors a_l satisfying F_i and some vectors b_s satisfying the homogeneous part of F_i it holds that

$$u = \sum_{l \in I_0} \left(a_l + \sum_{(l,s) \in J} \lambda_{ls} b_{ls} \right) + \sum_{l \in I_1} \mu_l a_l \wedge x_i = |I_0| + \sum_{l \in I_1} \mu_l$$

and furthermore (u, v) satisfies F_0. It follows that $F_0 \wedge \bigwedge_{i=1}^{n} u \in \{x \mid F_i(x)\}^{x_i}$ is satisfied by such u and v. Thus, $x \in \mathsf{LIA}^{card}$. □

8. Updated array theories classification

Traditionally, research on decision procedures for array theories has focused on providing concrete algorithms for syntactically presented theories. As a result, works on, in particular, combinatory array logic, are often unaware of each other which results in duplicated engineering work and the lack of a comparison between potentially different implementation techniques. Meseguer [43,44] has also expressed similar remarks about syntactically presented theories. In particular, in [43, Lecture 19], he notes:

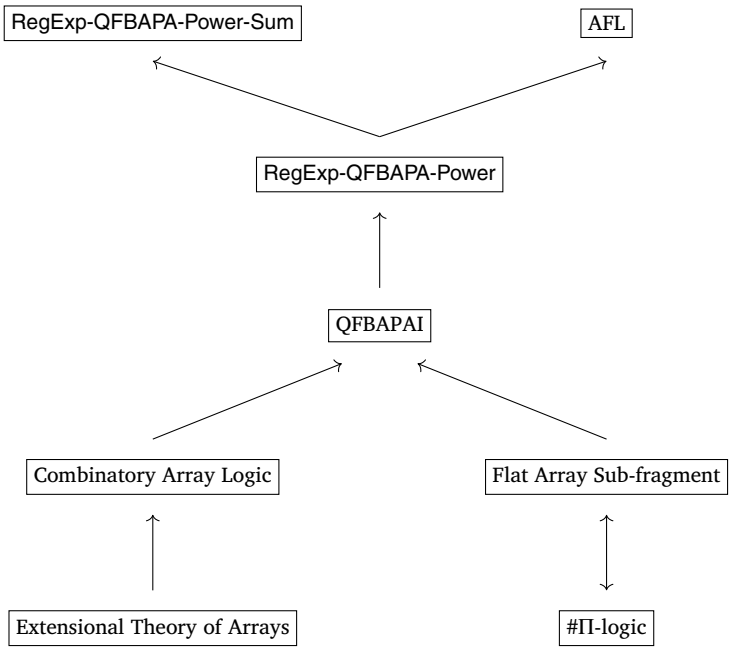


Fig. 6. Treated theories of arrays sorted by expressivity.

> The limitations of a syntactic notion of theory (...) are felt particularly strongly in the case of many decision procedures for satisfiability. In some cases the problem does not arise, as for the theories (...) where we are interested in all the models. They are however strongly felt when we want to obtain decision procedures for various data structures used in Computer Science or, more generally, any models used in Mathematics. In such cases, a syntactic approach to theories as the one taken, for example, in the Bradley and Manna book seems unreal, since:
>
> (i) it fails to explain what is the intended model (ii) knowledge of such intended model either cannot be used or has to be smuggled with some hand-waving, and (iii) the syntactic approach can easily hide from view the unsatisfactory fact that the syntactic theory (...) may after all fail to capture the intuitively intended model or models for which it was deployed in the first place. I will illustrate this fact with examples such as arrays and tables.

One advantage of our semantic approach is that one has a clear comparison criteria between the different theories proposed in the literature. Fig. 6 lists some related theories of arrays in the literature, ordered by expressivity. The satisfiability problem of these logics is in NP for all but the array fold logic (AFL) fragment, which in general lies in PSPACE. Note however, that RegExp-QFBAPA-Power would lead to PSPACE-hard problems when doing the conjunction of an arbitrary number of terms in the logic, by reduction from the intersection non-emptiness problem. Thus, this rises the question of whether array fold logic can be improved to support an arbitrary number of array variables in fold expressions. We leave this direction open to future work.

An important characteristic of these theories is the intended interpretation of the index and element set theories. As presented in [25,2] it could seem that the flat array sub-fragment and the Π-logic theory are restricted to integer indices or elements. The results on QFBAPAI show that this restriction is not necessary. However, RegExp-QFBAPA-Power already requires the index set to be the integers numbers and RegExp-QFBAPA-Power-Sum and AFL requires both the index and element set to be the integers. Note that we do not include in the hierarchy, SymRegExp-QFBAPA-Power since this theory has the same expressive power as that of RegExp-QFBAPA-Power.

It appears that sum constraints were of interest to the authors of [13] since the repository of the tool AFOLDER [50] contains some examples using sums. However, array fold logic as presented in [13] cannot express sums, since counters can only be updated by adding constants. Thus, to the best of our knowledge, the theory of arrays that we present in this paper is the first array theory whose satisfiability problem is decidable in non-deterministic polynomial time and can express sum constraints.

Finally, it is worth noting that there has been recent progress in the automation of linear arithmetic with stars [40, 53]. An adaptation of those techniques could be a good starting point for the implementation of decision procedures for RegExp-QFBAPA-Power-Sum.

9. Examples

In this section, we give examples of the expressible properties in the theory RegExp-QFBAPA-Power-Sum. Properties like boundedness, partitioning, periodicity, pumping, or equal count, which were taken from the examples presented in array folds logic [13],

were already encoded in [59]. In the next example, we show that the remaining properties given as example in [13] can also be encoded in the theory.

Example 50 *(AFL example translations).*

- Histogram. The histogram of the input data in the array a satisfies the distribution $H(\{i|a[i] < 10\} \geq 2H(\{i|a[i] \geq 10\})$.

$$|\{n \in \mathbb{N}|a[n] < 10\}| \geq 2|\{n \in \mathbb{N}|a[n] \geq 10\}|$$

- Length of Format Fields. The array contains two variable-length fields. The first two elements of the array define the length of each field; they are followed by the fields themselves, separated by 0.

$$\begin{aligned}
&\exists \bar{t} \in \binom{1}{0}\binom{0}{1}\binom{1}{1}^*\binom{0}{0}\binom{1}{0}^* \\
&\quad \wedge S_1 = \{n \in \mathbb{N} \mid (t_1 \wedge \neg t_2 \wedge \neg t_3)(n)\} \subseteq \{n \in \mathbb{N} \mid a[n] = \text{len}_1\} \\
&\quad \wedge S_2 = \{n \in \mathbb{N} \mid (\neg t_1 \wedge t_2 \wedge \neg t_3)(n)\} \subseteq \{n \in \mathbb{N} \mid a[n] = \text{len}_2\} \\
&\quad \wedge S_3 = \{n \in \mathbb{N} \mid (t_1 \wedge t_2 \wedge \neg t_3)(n)\} \wedge |S_3| = \text{len}_1 \wedge \\
&\quad \wedge S_4 = \{n \in \mathbb{N} \mid (\neg t_1 \wedge \neg t_2 \wedge t_3)(n)\} \subseteq \{n \in \mathbb{N} \mid a[n] = 0\} \\
&\quad \wedge S_5 = \{n \in \mathbb{N} \mid (t_1 \wedge \neg t_2 \wedge t_3)(n)\} \wedge |S_5| = \text{len}_2 \\
&\quad \wedge (S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5)^c = \{n \in \mathbb{N} \mid a[n] = \#\}
\end{aligned}$$

We have already noted that summation constraints are not expressible in [13]. Below, we give examples that use aggregation constraints, including summations.

Example 51 *(Aggregation constraints).*

- Given an array, accept it if the number of minimum elements in the array is the same as the number of maximum elements in the array.

$$\begin{aligned}
&\exists \bar{t} \in 1^* \wedge S = \{n \in \mathbb{N}|t(n)\} = \{n \in \mathbb{N}|a[n] \neq \#\} \wedge \\
&\qquad |\{n \in \mathbb{N}|a[n] = \max(a)\}| = |\{n \in \mathbb{N}|a[n] = \min(a)\}|
\end{aligned}$$

- There is a pair of elements in the array that sum exactly to targetSum.

$$\exists \bar{t} \in \binom{1}{0}^*\binom{0}{1}\binom{1}{0}^*\binom{0}{1}\binom{1}{0}^* \wedge \sum_{n \in \mathbb{N}} (|\bar{c}(n)|(\neg t_1 \wedge t_2)(n)|) = \text{targetSum}$$

- The average of the elements in an array is a given number n.

$$\begin{aligned}
&\exists t \in 1^* \wedge S = \{n \in \mathbb{N}|t(n)\} = \{n \in \mathbb{N}|a[n] \neq \#\} \\
&\qquad \wedge \sigma = \sum_{n \in \mathbb{N}} (|c(n)|a[n] \neq \#|) \wedge \sigma = n|S|
\end{aligned}$$

- The sum of the elements of the array should be divisible by three.

$$\begin{aligned}
&\exists t \in 1^* \wedge S = \{n \in \mathbb{N}|t(n)\} = \{n \in \mathbb{N}|a[n] \neq \#\} \\
&\qquad \wedge \sigma = \sum_{n \in \mathbb{N}} (|c(n)|a[n] \neq \#|) \wedge 3|\sigma
\end{aligned}$$

10. Conclusion

We have contributed to the study of array theories that can be expressed using power structures as a basis. We observed that the introduction of an ordering relation can be done succinctly using recent results on the Parikh image of symbolic automata, which we showed have a simple explanation in the framework of the Feferman-Vaught theorem: the computation of the Parikh image reduces to counting monadic formulas and adding a constraint stating that certain subsets of the universe form a partition.

Then, we moved our study to express aggregation functions, which are of crucial importance in applications in verification and databases. We showed that our extension of combinatory array logic (i.e. array theory with vector-like operations) can express minimum and maximum of indices on the arrays as well as minimum and maximum of the elements contained in the arrays. We also showed that the summation operator is supported by the fragment.

Unlike in other recent logics for arrays like array folds logic [13], both developments are expressible by a logic that does not restrict the number of array variables and whose satisfiability problem is NP-complete. This encourages us to, in future work, start

implementing the framework in SMT solvers, perhaps reusing the existing solvers for BAPA [67,5] and QFBAPAI [2], as well as software computing Parikh images [27]. As an intermediate step, we plan to already implement the method for expressing the succinct ordering relation, which can be considered an optimisation for the problem of deciding non-emptiness of symbolic finite automata. In the longer term, we would like to apply the decomposition methodology of Feferman-Vaught to other array theories, such as the classical [8]. Are there reasonable decompositions for other fragments of the theory of arrays in the literature?

CRediT authorship contribution statement

Rodrigo Raya: Writing – original draft, Investigation. **Viktor Kunčak:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Elvira Albert, Jesús Correas, Pablo Gordillo, Guillermo Román-Díez, Albert Rubio, GASOL: gas analysis and optimization for Ethereum smart contracts, in: Tools and Algorithms for the Construction and Analysis of Systems, in: Series Title: Lecture Notes in Computer Science., vol. 12079, Springer International Publishing, Cham, 2020, pp. 118–125.
[2] F. Alberti, S. Ghilardi, E. Pagani, Cardinality constraints for arrays (decidability results and applications), Form. Methods Syst. Des. 51 (3) (December 2017) 545–574.
[3] Leonardo Alt, Martin Blicha, Antti E.J. Hyvärinen, Natasha Sharygina, SolCMC: solidity compiler's model checker, in: Sharon Shoham, Yakir Vizel (Eds.), Computer Aided Verification, in: Lecture Notes in Computer Science, Springer International Publishing, Cham, 2022, pp. 325–338.
[4] Sanjeev Arora, Boaz Barak, Computational Complexity: A Modern Approach, Cambridge University Press, Cambridge; New York, 2009.
[5] Kshitij Bansal, Andrew Reynolds, Clark Barrett, Cesare Tinelli, A new decision procedure for finite sets and cardinality constraints in SMT, in: Automated Reasoning, in: Lecture Notes in Computer Science, Springer International Publishing, Cham, 2016, pp. 82–98.
[6] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, Yoni Zohar, cvc5: a versatile and industrial-strength SMT solver, in: Dana Fisman, Grigore Rosu (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, in: Lecture Notes in Computer Science, Springer International Publishing, Cham, 2022, pp. 415–442.
[7] Aaron Bradley, Zohar Manna, Calculus of computation: decision procedures with applications to verification, Springer, Berlin, 2007, OCLC, ocn190764844.
[8] Aaron R. Bradley, Zohar Manna, Henny B. Sipma, What's decidable about arrays?, in: Verification, Model Checking, and Abstract Interpretation, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2006, pp. 427–442.
[9] Stanley N. Burris, Logic for Mathematics and Computer Science, 1st edition, Prentice Hall, Upper Saddle River, N.J, August 1997.
[10] J.R. Büchi, D. Siefkes, Decidable Theories: Vol. 2: The Monadic Second Order Theory of All Countable Ordinals, Springer, Berlin, 1973.
[11] J. Richard Büchi, Weak second-order arithmetic and finite automata, Math. Log. Q. 6 (1–6) (1960) 66–92.
[12] Hubert Comon, Max Dauchet, Remi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Loding, Sophie Tison, Marc Tommasi, Tree Automata Techniques and Applications, INRIA, 2008.
[13] Przemysław Daca, Thomas A. Henzinger, Andrey Kupriyanov, Array folds logic, in: Computer Aided Verification, in: Lecture Notes in Computer Science, Springer International Publishing, Cham, 2016, pp. 230–248.
[14] Loris D'Antoni, Margus Veanes, Minimization of symbolic tree automata, in: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, ACM, New York NY USA, July 2016, pp. 873–882.
[15] Loris D'Antoni, Margus Veanes, Benjamin Livshits, David Molnar, Fast: a transducer-based language for tree manipulation, ACM Trans. Program. Lang. Syst. 38 (1) (October 2015) 1:1–1:32.
[16] Leonardo de Moura, Nikolaj Bjorner, Generalized efficient array decision procedures, in: 2009 Formal Methods in Computer-Aided Design, IEEE, Austin, TX, November 2009, pp. 45–52.
[17] Leonardo de Moura, Nikolaj Bjørner, Z3: an efficient SMT solver, in: Tools and Algorithms for the Construction and Analysis of Systems, in: Lecture Notes in Computer Science, vol. 4963, Springer, Berlin, Heidelberg, 2008, pp. 337–340.
[18] John Doner, Tree acceptors and some of their applications, J. Comput. Syst. Sci. 4 (5) (October 1970) 406–451.
[19] Andrzej Ehrenfeucht, An application of games to the completeness problem for formalized theories, Fundam. Math. 49 (2) (1961) 129–141.
[20] Friedrich Eisenbrand, Gennady Shmonin, Carathéodory bounds for integer cones, Oper. Res. Lett. 34 (5) (September 2006) 564–568.
[21] Neta Elad, Sophie Rain, Neil Immerman, Laura Kovács, Mooly Sagiv, Summing up smart transitions, in: Alexandra Silva, K. Rustan, M. Leino (Eds.), Computer Aided Verification, in: Lecture Notes in Computer Science, Springer International Publishing, Cham, 2021, pp. 317–340.
[22] S. Feferman, R. Vaught, The first order properties of products of algebraic systems, Fundam. Math. 47 (1) (1959) 57–103.
[23] Jeanne Ferrante, Charles Rackoff, The Computational Complexity of Logical Theories, Springer, Berlin, Heidelberg, 1979.
[24] Zoltán Fülöp, Heiko Vogler, Forward and backward application of symbolic tree transducers, Acta Inform. 51 (5) (August 2014) 297–325.
[25] Klaus v. Gleissenthall, Nikolaj Bjørner, Andrey Rybalchenko, Cardinalities and universal quantifiers for verifying parameterized systems, in: Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '16, Association for Computing Machinery, New York, NY, USA, June 2016, pp. 599–613.
[26] Matthew Hague, Artur Jeż, Anthony W. Lin, Parikh's theorem made symbolic, Proc. ACM Program. Lang. 8 (POPL) (January 2024) 65:1945–65:1977.
[27] Matthew Hague, Anthony Widjaja Lin, Synchronisation - reversal-bounded analysis of multithreaded programs with counters, in: P. Madhusudan, Sanjit A. Seshia (Eds.), Computer Aided Verification, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2012, pp. 260–276.
[28] Lauri Hella, Leonid Libkin, Juha Nurmonen, Limsoon Wong, Logics with aggregate operators, J. ACM 48 (4) (July 2001) 880–907.
[29] Wilfrid Hodges, Model Theory, Encyclopedia of Mathematics and Its Applications, Cambridge University Press, Cambridge, 1993.
[30] Gérard Huet, Initiation à la Logique Mathématique, 1986.

[31] James Cornelius King, A program verifier, PhD thesis, Carnegie-Mellon University, Pittsburgh Pennsylvania USA, September 1969, Section: Technical Reports.
[32] Michael Kaminski, Tony Tan, Tree automata over infinite alphabets, in: Pillars of Computer Science: Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2008, pp. 386–423.
[33] Felix Klaedtke, Harald Rueß Parikh, Automata and Monadic Second-Order Logics with Linear Cardinality Constraints, Technical Report 177, Freiburg University, Institute of Computer Science, 2002.
[34] S.C. Kleene, Representation of events in nerve nets and finite automata, in: Representation of Events in Nerve Nets and Finite Automata, Princeton University Press, 1956, pp. 3–42.
[35] Daniel Kroening, Ofer Strichman, Decision Procedures, 2 edition, Texts in Theoretical Computer Science. An EATCS Series, Springer, Berlin, Heidelberg, 2016.
[36] Viktor Kunčak, Huu Hai Nguyen, Martin Rinard, Deciding Boolean algebra with Presburger arithmetic, J. Autom. Reason. 36 (3) (April 2006) 213–239.
[37] Viktor Kunčak, Ruzica Piskac, Philippe Suter, ordered sets in the calculus of data structures, in: Computer Science Logic, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2010, pp. 34–48.
[38] Kunčak Viktor, Martin Rinard, Towards efficient satisfiability checking for Boolean algebra with Presburger arithmetic, in: Automated Deduction – CADE-21, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2007, pp. 215–230.
[39] Rustan Leino, Dafny: an automatic program verifier for functional correctness, in: Logic for Programming, Artificial Intelligence, and Reasoning, April 2010, pp. 348–370.
[40] Maxwell Levatich, Nikolaj Bjørner, Ruzica Piskac, Sharon Shoham, Solving LIA* using approximations, in: Dirk Beyer, Damien Zufferey (Eds.), Verification, Model Checking, and Abstract Interpretation, in: Lecture Notes in Computer Science, Springer International Publishing, Cham, 2020, pp. 360–378.
[41] Denis Lugiez, Silvano Dal Zilio, Multitrees Automata, Presburger's Constraints and Tree Logics, Technical Report 08-2002, Laboratoire d'Informatique Fondamentale de Marseille, 2002.
[42] Yuri Matiyasevich, Hilbert's 10th Problem. Foundations of Computing, MIT Press, 1993.
[43] José Meseguer, Lecture Notes on Topics in Automated Reasoning, 2017.
[44] José Meseguer, Variants and satisfiability in the infinitary unification wonderland, J. Log. Algebraic Methods Program. 134 (August 2023) 100877.
[45] Andrzej Mostowski, On direct products of theories, J. Symb. Log. 17 (1) (March 1952) 1–31.
[46] Greg Nelson, Derek C. Oppen, Simplification by cooperating decision procedures, ACM Trans. Program. Lang. Syst. 1 (2) (October 1979) 245–257.
[47] Rodrigo Otoni, Matteo Marescotti, Leonardo Alt, Patrick Eugster, Antti Hyvärinen, Natasha Sharygina, A solicitous approach to smart contract verification, ACM Trans. Priv. Secur. 26 (2) (March 2023) 15:1–15:28.
[48] Christos H. Papadimitriou, On the complexity of integer programming, J. ACM 28 (4) (October 1981) 765–768.
[49] R.J. Parikh, Language generating devices, Technical Report, Research Laboratory of Electronics (RLE) at the Massachusetts Institute of Technology (MIT), January 1961, Accepted: 2010-04-01T22:52:42Z.
[50] pdaca. AFolder - Solver for Array Folds Logic., October 2022, original-date: 2016-03-22T15:29:30Z.
[51] Anton Permenev, Dimitar Dimitrov, Petar Tsankov, Dana Drachsler-Cohen, Martin Vechev, VerX: safety verification of smart contracts, in: 2020 IEEE Symposium on Security and Privacy (SP), ISSN 2375-1207, May 2020, pp. 1661–1677.
[52] Ruzica Piskac, Decision Procedures for Program Synthesis and Verification, PhD thesis, EPFL, Lausanne, 2011.
[53] Ruzica Piskac, Efficient automated reasoning about sets and multisets with cardinality constraints, in: Automated Reasoning, in: Lecture Notes in Computer Science, Springer International Publishing, Cham, 2020, pp. 3–10.
[54] Ruzica Piskac, Viktor Kunčak, Linear arithmetic with stars, in: Computer Aided Verification, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2008, pp. 268–280.
[55] Ruzica Piskac, Thomas Wies, Decision procedures for automating termination proofs, in: Verification, Model Checking, and Abstract Interpretation, in: Lecture Notes in Computer Science, vol. 6538, Springer, Berlin, Heidelberg, 2011, pp. 371–386.
[56] Rodrigo Raya, The complexity of checking non-emptiness in symbolic tree automata, arXiv:2311.05250 [cs], November 2023.
[57] Rodrigo Raya, The complexity of satisfiability checking for symbolic finite automata, arXiv:2307.00151 [cs], June 2023.
[58] Rodrigo Raya, Viktor Kunčak, NP satisfiability for arrays as powers, in: Verification, Model Checking, and Abstract Interpretation, in: Lecture Notes in Computer Science, Springer International Publishing, Cham, 2022, pp. 301–318.
[59] Rodrigo Raya, Viktor Kunčak, On algebraic array theories, J. Log. Algebraic Methods Program. 136 (January 2024) 100906.
[60] Helmut Seidl, Thomas Schwentick, Anca Muscholl, Peter Habermehl, Counting in trees for free, in: Automata, Languages and Programming, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2004, pp. 1136–1149.
[61] Joseph Shoenfield, Mathematical Logic, Addison-Wesley Publishing Company, Inc., 1967.
[62] Michael Sipser, Introduction to the Theory of Computation. Cengage Learning, 3 edition, June 2012.
[63] Thoralf Skolem, Untersuchungen über die Axiome des Klassenkalküls und über Produktations- und Summationsprobleme, welche gewisse Klassen von Aussagen betreffen, 1919.
[64] Solomon Feferman, Product operations on relational systems (abstract), Bull. Am. Math. Soc. 61 (172) (March 1955).
[65] Larry Stockmeyer, Albert R. Meyer, Cosmological lower bound on the circuit complexity of a small problem in logic, J. ACM 49 (6) (November 2002) 753–784.
[66] A. Stump, C.W. Barrett, D.L. Dill, J. Levitt, A decision procedure for an extensional theory of arrays, in: Proceedings 16th Annual IEEE Symposium on Logic in Computer Science, IEEE Comput. Soc., Boston, MA, USA, 2001, pp. 29–37.
[67] Philippe Suter, Robin Steiger, Viktor Kunčak, Sets with cardinality constraints in satisfiability modulo theories, in: Verification, Model Checking, and Abstract Interpretation, in: Lecture Notes in Computer Science, vol. 6538, Springer, Berlin, Heidelberg, 2011, pp. 403–418.
[68] Nikhil Swamy, Juan Chen, Ben Livshits, Verifying higher-order programs with the Dijkstra monad, in: Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation, Seattle, Washington, USA, June 2013.
[69] Wolfgang Thomas, Ehrenfeucht, Vaught, and the decidability of the weak monadic theory of successor, ACM SIGLOG News 5 (1) (January 2018) 14–18.
[70] Niki Vazou, Eric L. Seidel, Ranjit Jhala, Dimitrios Vytiniotis, Simon Peyton-Jones, Refinement types for Haskell, in: Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming, ICFP '14, Association for Computing Machinery, New York, NY, USA, August 2014, pp. 269–282.
[71] Margus Veanes, Nikolaj Bjørner, Symbolic tree automata, Inf. Process. Lett. 115 (3) (March 2015) 418–424.
[72] Margus Veanes, Nikolaj Bjørner, Leonardo de Moura, Symbolic automata constraint solving, in: Christian G. Fermüller, Andrei Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2010, pp. 640–654.
[73] Kumar Neeraj Verma, Helmut Seidl, Thomas Schwentick, On the Complexity of Equational Horn Clauses, Automated Deduction – CADE-20, vol. 3632, Springer, Berlin, Heidelberg, 2005, pp. 337–352.
[74] Nicolas Charles Yves Voirol, Verified Functional Programming, PhD thesis, EPFL, Lausanne, 2019.
[75] Thomas Wies, Ruzica Piskac, Viktor Kunčak, Combining theories with shared set operations, in: Frontiers of Combining Systems, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2009, pp. 366–382.